12

AD A119 [illegible]

RADC-TR-82-161
Final Technical Report
June 1982

# ADVANCED IMAGING TRACKER

Adaptive Optics Associates, Inc.

Sponsored by
Defense Advanced Research Projects Agency (DOD)
ARPA Order No. 3503

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

RADC-TR-82-161 has been reviewed and is approved for publication.

APPROVED: [illegible]

PATRICK J. MARTONE, Capt, USAF
Project Engineer

APPROVED: [illegible]

LOUIS E. WHITE, Lt Col, USAF
Acting Chief
Surveillance Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

# ADVANCED IMAGING TRACKER

Dr. L. E. Schmutz

Contractor: Adaptive Optics Associates, Inc.
Contract Number: F30602-80-C-0216
Effective Date of Contract: 23 June 1980
Contract Expiration Date: 31 January 1982
Short Title of Work: Advanced Imaging Tracker
Program Code Number: 1E20
Period of Work Covered: Jun 80 - Dec 81

Principal Investigator: Dr. Larry Schmutz
Phone: 617 547-2786

Project Engineer: Captain Patrick J. Martone
Phone: 315 330-3145

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Capt Patrick J. Martone (RADC/OCSE), Griffiss AFB NY 13441 under Contract F30602-80-C-0216.

X

A

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER: RADC-TR-82-161 | 2. GOVT ACCESSION NO.: AD-A119045 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle): ADVANCED IMAGING TRACKER | | 5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report 10 Jun 80 - 31 Dec 81 |
| | | 6. PERFORMING ORG. REPORT NUMBER: N/A |
| 7. AUTHOR(s): Dr. L. E. Schmutz | | 8. CONTRACT OR GRANT NUMBER(s): F30602-80-C-0216 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Adaptive Optics Associates, Inc. 2336 Massachusetts Avenue Cambridge MA 02140 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62301E C5030106 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency 1400 Wilson Blvd Arlington VA 22209 | | 12. REPORT DATE: June 1982 |
| | | 13. NUMBER OF PAGES: 186 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Rome Air Development Center (OCSE) Griffiss AFB NY 13441 | | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Patrick J. Martone, Capt, USAF (OCSE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | |
|---|---|
| Imaging | Image Processing |
| Tracking | Digital Signal Processing |
| Infrared | Quad Cell |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A new imaging-tracker device capable of tracking a distant optical or infrared target at response rates of 8 KHz, and imaging that target at frame rates of 30 Hz to 8x8 pixel resolution has been developed and demonstrated in computer simulation and in hardware implementation. The system uses a simple quad cell detector for both tracking and imaging, employing an advanced scanner system and reconstruction technique to perform imaging. Image reconstruction and centroid tracking algorithms have been defined

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

and characterized analytically and numerically in terms of accuracy and signal-to-noise performance. A dedicated high-speed digital processor system was developed and used to implement the imaging and tracking algorithms in conjunction with a specially designed optical breadboard of the imager-tracker system.

## SUMMARY

This technical report describes the results of a two-year investigation into the Advanced Imaging Tracker (AIT), a novel infrared tracking concept which attempts to provide low resolution imagery at frame rates comparable to contemporary FLIR's with simultaneous high-bandwidth target centroid estimation. The initial motivation was to find useful alternative passive tracking systems for the DARPA Talon Gold and LODE programs.

The program was divided into two phases. Phase I, a concept feasibility study, was completed in December 1980. The results of this phase were combined in an Interim Report. The Phase II program effort centered on fabricating a working AIT portable breadboard, containing all optics, electronics, processing hardware and software needed for conducting an AIT experimental test program.

The breadboard was constructed and AIT imaging and tracking demonstrated. During Phase II new classes of imaging and tracking algorithms were defined, analyzed, characterized by computer simulation and implemented in hardware. A special purpose high-speed digital signal processing system, the Programmable Microcoded Processor (PMP), was designed, fabricated, tested and coded. The optical system is capable of visible and IR operation, and includes a dynamic target simulator for system testing. An extensive software base was written to support algorithm development, system simulation, experimental data analysis, and interactive control. A color display system was integrated with the breadboard system for pseudo-color image output and diagnostic display. The system is ready for use in an experimental program of AIT imaging and tracking performance.

TABLE OF CONTENTS

Page

1.0 INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . 1
1.1 AIT Concept . . . . . . . . . . . . . . . . . . . . . 1
1.2 Aim-Point Maintenance . . . . . . . . . . . . . . . 4
1.3 AIT Program Goals . . . . . . . . . . . . . . . . . 7

2.0 THEORETICAL FORMULATION: IMAGING . . . . . . . . . . 9
2.1 The $I^3$ Sensor . . . . . . . . . . . . . . . . . . 9
2.2 Nutation of Extended Objects . . . . . . . . . . 12
2.3 The AIT Image Reconstruction Algorithm . . . . 18
2.4 Properties of the Reconstruction Algorithm . . 21

3.0 THEORETICAL FORMULATION: TRACKING . . . . . . . . . . 25
3.1 Algorithm Constraints . . . . . . . . . . . . . . 25
3.2 The SCS Algorithm . . . . . . . . . . . . . . . . 28
3.3 The AIT Tracking Algorithm . . . . . . . . . . . 33
3.4 AIT Tracker S/N . . . . . . . . . . . . . . . . . 40

4.0 SOFTWARE SIMULATION AND PERFORMANCE CHARACTERIZATION: IMAGING . . . . . . . . . . . . . . 42
4.1 Computer System . . . . . . . . . . . . . . . . . 42
4.2 Simulation Structure . . . . . . . . . . . . . . . 45
4.3 Generation of the Back Matrix . . . . . . . . . 48
4.4 Image Quality . . . . . . . . . . . . . . . . . . 51
4.5 Evaluation of Image Noise . . . . . . . . . . . . 53
4.6 Simulation Results . . . . . . . . . . . . . . . . 58
4.7 Algorithm Parameter Sensitivity . . . . . . . . 64

5.0 SOFTWARE SIMULATION: TRACKING . . . . . . . . . . . . 71
5.1 Tracking Performance . . . . . . . . . . . . . . . 77

6.0 BREADBOARD HARDWARE: OPTICAL SYSTEM . . . . . . . . . 81
6.1 Optical Layout . . . . . . . . . . . . . . . . . . 81
6.2 Nutation . . . . . . . . . . . . . . . . . . . . . 86
6.3 Interface Electronics . . . . . . . . . . . . . . 97

TABLE OF CONTENTS (continued)

Page

7.0 BREADBOARD HARDWARE: ELECTRONIC PROCESSOR . . . . 99

7.1 Processor Requirements . . . . . . . . . . . . 99

7.2 PMP Architecture . . . . . . . . . . . . . . . . 100

7.3 AIT Microprogram Structure . . . . . . . . . . 104

7.4 Processor Performance . . . . . . . . . . . . 111

7.5 Vector Matrix Multiplier . . . . . . . . . . . 116

8.0 EXPERIMENTAL SOFTWARE . . . . . . . . . . . . . . . 120

8.1 PMP Support . . . . . . . . . . . . . . . . . . 121

8.2 Bench Support . . . . . . . . . . . . . . . . . 124

8.3 The Genisco Color Display . . . . . . . . . . 126

8.4 General Disk File Handling Utilities . . . . . 128

9.0 EXPERIMENTAL BREADBOARD RESULTS . . . . . . . . . . 129

10.0 SUMMARY AND CONCLUSIONS . . . . . . . . . . . . . . 136

REFERENCES . . . . . . . . . . . . . . . . . . . . . . . 139

APPENDIX I: SOFTWARE SUMMARY . . . . . . . . . . . . . 140

## 1.0 INTRODUCTION

### 1.1 AIT Concept

The Advanced Imaging Tracker (AIT) program undertook to develop a novel tracker-imager concept. The scheme employs a quad-cell detector and an image scanning device to extract from a received optical image its centroid and a low-resolution reconstruction of the image. The system was originally intended as an advance on scanning-FLIR type imaging trackers, since it could provide a much higher centroid tracking rate (4-8 kHz, as opposed to 30 Hz for video-rate FLIR's) while retaining the FLIR image rate (30 Hz).

The AIT concept is described in somewhat more detail in Figure 1.1. Passive infrared radiation (dashed line) from a distant target is received by a telescope and transmitted to the AIT through some optical relay system including, in this case, a set of fast tilt mirrors for fine track on the target. In this example, the telescope is part of an HEL beam director system, so that a Shared Aperture Device is included in the optical path.

The optical layout of the AIT itself is shown figuratively at bottom left in the diagram. Here the major components of the AIT are shown. A carefully prepared optical plane wave (solid), whose amplitude may be controlled by a modulator, is combined with the incoming IR from the telescope, and reflected into the remaining tracker optics. The plane wave is used as a highly accurate tilt reference, which may be used for differential measurement of the image centroid in the focal plane, removing measurement errors

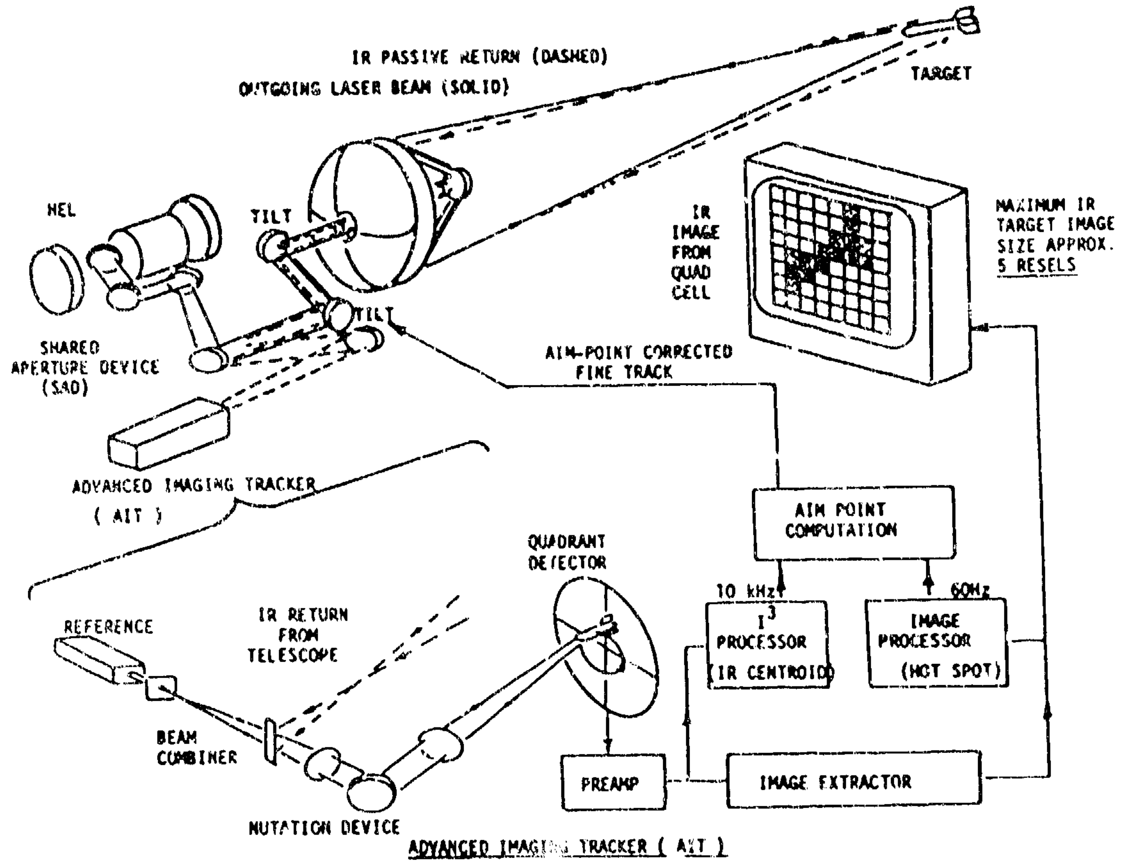

IR PASSIVE RETURN (DASHED)
OUTGOING LASER BEAM (SOLID)
TARGET
HEL
TILT
TILT
IR IMAGE FROM QUAD CELL
MAXIMUM IR TARGET IMAGE SIZE APPROX. 5 RESELS
SHARED APERTURE DEVICE (SAD)
AIM-POINT CORRECTED FINE TRACK
ADVANCED IMAGING TRACKER ( AIT )
AIM POINT COMPUTATION
QUADRANT DETECTOR
10 kHz
60Hz
REFERENCE
IR RETURN FROM TELESCOPE
I3 PROCESSOR (IR CENTROID)
IMAGE PROCESSOR (HOT SPOT)
BEAM COMBINER
PREAMP
IMAGE EXTRACTOR
NUTATION DEVICE
ADVANCED IMAGING TRACKER ( AIT )

FIGURE 1.1

induced by motion of the AIT optical components, electronics offsets, etc.

The combined beam is passed through a high speed nutation device, depicted here as a tip-tilt mirror. This device deflects the beam in a complex pattern in the focal plane, where it is detected by a simple quadrant detector. The result is a set of four output wavefroms, one from each detector, which are passed to a pair of signal processors.

The first of these is based on the centroid extraction technique (the Synchronous Centroid Sampling, or SCS, algorithm) which was developed by AOA for the $I^3$ Sensor wavefront sensor. The details of this process will be discussed in Section 3, but its purpose is to determine the IR centroid of the image, and hence angular position of the target object.

The second processor is used to reconstruct the target image from the detector wavefroms. The image information, obtained at relatively slow speed, can then be combined with the high speed centroid data to determine the appropriate aim point for the HEL beam on the target object. The computed aim point results are then applied to the tip-tilt mirrors to maintain system track.

The AIT process therefore consists of three basic steps:

1) Nutate the image on a quadrant detector to produce the output waveforms which carry the centroid and image information.

2) Process the waveforms for centroid data.

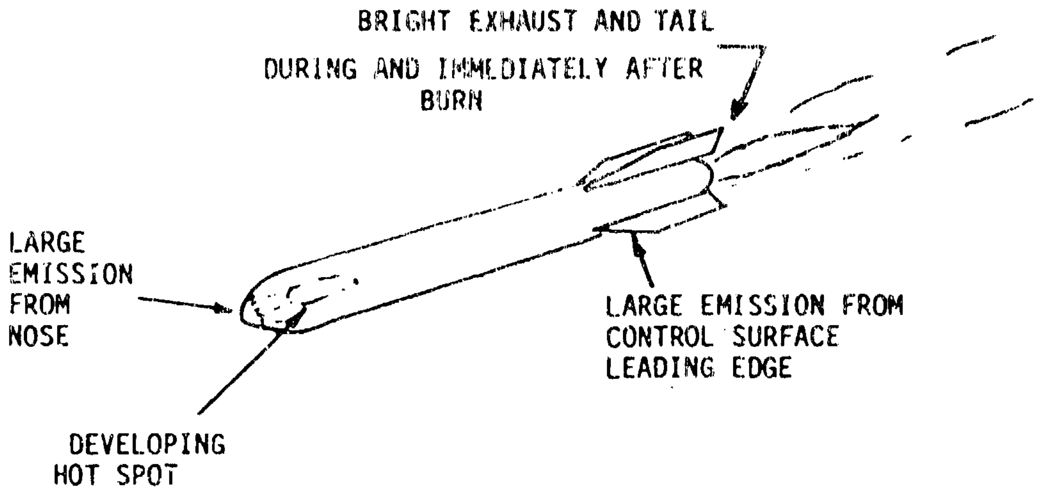
3) Process the waveforms for image reconstruction.

## 1.2 Aim-Point Maintenance

The need for both image and centroid data is outlined in Figure 1.2. Whereas the centroid of the image may be obtained at high speed (at least using the AIT technique) it does not provide sufficient information to stabilize an HEL beam on a relatively small region of an extended target. This is because the target will in general be thermally dynamic; that is, its temperature profile will change in time due both to its intrinsic characteristics and its interaction with the impinging HEL beam. In the example of Figure 1.2, the target is an atmospheric missile. Due to air frictional heating and the different thrust phases of its flight, the temperature distribution on the target will change significantly as the leading edges heat up, the engine area warms, and the exhaust plume vanishes at burnout.

Besides these intrinsic effects, the target will develop a hot spot at the point of initial contact with the HEL beam. For a tracker system in which the HEL aim point is not coincident with the centroid measurement, the contact point would be offset from the centroid of the IR emission of the target alone. As the hot spot develops, however, the centroid will be shifted towards the hot spot, which will in turn change the aim point. This interaction will cause the hot spot to drift on the object, greatly reducing the effectiveness of the HEL.

If information from the full image is available in addition to its centroid, the HEL aimpoint may be calculated as an offset to the IR centroid. This offset can than be updated as the target


BRIGHT EXHAUST AND TAIL
DURING AND IMMEDIATELY AFTER BURN
LARGE EMISSION FROM NOSE
LARGE EMISSION FROM CONTROL SURFACE LEADING EDGE
DEVELOPING HOT SPOT

AIM POINT MAINTENANCE IS REQUIRED BECAUSE IR CENTROID CHANGES POSITION ON OBJECT DURING DIFFERENT PHASES OF FLIGHT AND AS HOT SPOT DEVELOPS. TO KEEP HOT SPOT STABLE ON OBJECT, AIM POINT OFFSETS FROM IR CENTROID CAN BE CALCULATED FROM IR IMAGE INFORMATION.

FIGURE 1.2

temperature profile changes, stabilizing the HEL aimpoint. Since the temperature profile is governed by thermal time constants, it vaires slowly, on the order of tens of milliseconds, compared with the sub-millisecond jitter rates of the centroid itself (due to effects such as atmospheric disturbance and telescope bearing jitter).

The AIT is therefore intended to yield the combination of fast centroid tracking and slower image reconstruction (at a ratio of about 100:1) required for aim-point maintenance.

## 1.3 AIT Program Goals

The AIT program was designed to analyze the concept and identify the relevant theoretical and technological issues, characterize the problems and develop solutions, and finally implement the results in a working breadboard imager-tracker.

The problems of imaging and tracking were addressed separately. The tracking problem was treated as an extension of the $I^3$ Sensor measurement technique, and the basic system constraints imposed by the need to track were determined. The imaging problem was then analyzed subject to the tracking constraints, and a formal procedure for obtaining image reconstruction developed. The tracking and imaging algorithms were then evaluated by extensive computer simulation to verify the analytical results. A processing architecture capable of implementing the algorithms was adopted, and a complete processor designed and fabricated, and coded to execute both imaging and tracking processes. A breadboard version of the optical sensor head itself was then constructed, and integrated with a specially built target simulator to test the performance of a complete system. Finally, operation of all components was demonstrated, and image reconstruction experimentally achieved.

In the following sections each of the major areas of program effort are described and results presented. The material is organized topically rather than chronologically, since many of the efforts were parallel; the time sequence implied by the section numbers is however roughly correct.

The AIT program was divided into two parts: a Phase I feasibility study which extended from June 1980 to December 1980, and a Phase II breadboard fabrication and test program conducted from January 1981 to January 1982. The results of Phase I have been repeated and extended, so that this report documents the output of both phases.

## 2.0 THEORETICAL FORMULATION: IMAGING

### 2.1 The $I^3$ Sensor

The AIT tracker-imager is an extension of the $I^3$ Sensor(1) centroid tracker, and it is useful to first examine the simpler $I^3$ Sensor for insight into the operation of the AIT.

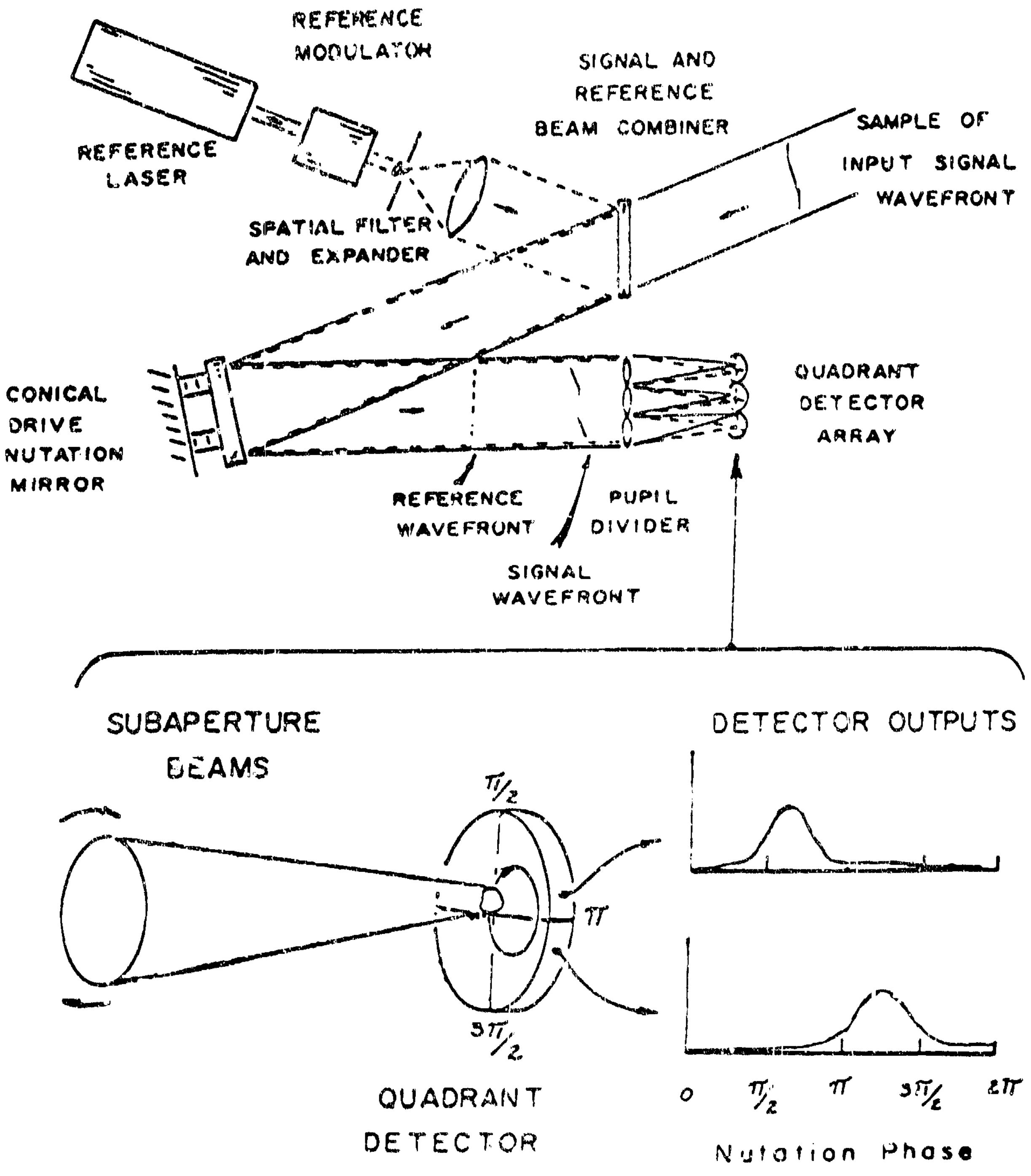
Figure 2.1 is a schematic view of the $I^3$ Sensor. Designed as an optical wavefront sensor, it is, in its simplest form, an array of tilt sensors, each measuring the mean tilt in a subaperture of the instrument's full input aperture. The resulting 2-D array of wavefront slope measurements can then be used to obtain an estimate of the original wavefront shape.

In operation, the input wavefront is combined with a plane wave which acts as a tilt reference. The reference is switchable, and is used only a small fraction of the time to calibrate the sensor for its own optical and electronic drifts. The wavefronts are then relayed through a nutation device, in the case of the $I^3$ sensor, which imposes a simple circular scan on the beams. The full aperture is then divided into subapertures by an array of sampling lenslets. At the focal plane of each lenslet is a quadrant detector.

The detector focal plane is shown enlarged at the bottom of the figure. As the subaperture focal spot moves in the circular pattern induced by the nutation scan, the detector quadrant outputs vary. The intensity in each quadrant increases as the spot moves

$I^3$ SENSOR*

Principles of Operation



*U.S. Patent No. 4,141,652

FIGURE 2.1

into that quadrant; at the same time the intensity in the adjacent quadrant decreases as the spot leaves. When appropriately demodulated, as described in Section 3, the centroid of the unnutated focal spot (or the centroid of the circular orbit) is obtained in each subaperture.

Taken individually, each subaperture tilt sensor is a complete centroid tracker. Under the Advanced Imaging Adaptive Optics (AIAO) program, it was shown that the variance $\sigma$ on the angular measurement is given by

$$\sigma^2 = \frac{2}{SNR} \left(\frac{\lambda}{D}\right)^2 \text{(Radians)}^2 \qquad (2.1)$$

where SNR = signal-to-noise ratio of the full four-quadrant detector area in the operating bandwidth

$\lambda$ = wavelength of incident radiation

D = the subaperture diameter

or, for photon-noise limited operation

$$\sigma^2 = \frac{2}{N} \left(\frac{\lambda}{D}\right)^2 \text{(radians)}^2 \qquad (2.2)$$

where N = number of photons detected during observation period

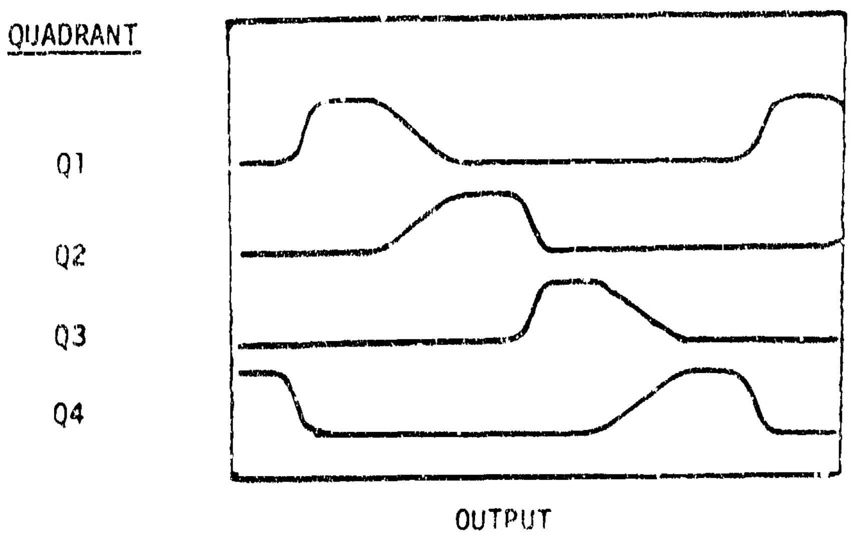
This is within a factor of two of the theoretical performance limit for quad-cell type detection,[2] so that the $I^3$ sensor is a near-optimal centroid tracker.

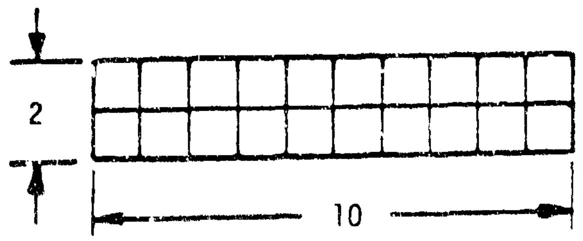
## 2.2 Nutation of Extended Objects

By examination of the detector waveforms for an extended object, it can be seen that more information than just centroid data has been encoded by the nutated quad-cell process. Figure 2.2 shows the quad cell outputs for a horizontal bar target having a 5:1 aspect ratio. The waveforms were experimentally measured on a breadboard sensor set-up. The oblong nature of the bar is apparent by the slope differences in the waveforms for, say, the Q1-Q2 transition and the Q2-Q3 transition. As the long axis of the bar crosses between quadrants the transition is gradual, while the transition is abrupt when the narrow axis moves across the quadrant boundary.

The nature of the transformation between image and nutation may be quantified by considering the object to be composed of pixel elements of varying intensity in the image plane. An illustration of the results of such a decomposition is shown in Figures 2.3 and 2.4. In Figure 2.3, three different pixels from a field of 8 x 8 are shown illuminated. Because of their differing displacements from the quad cell origin, the arcs described by each will result in different residence times for each pixel in the four quadrants. Simulated quadrant outputs for each pixel are shown in Figure 2.4. From these considerations one is invited to make a one-to-one correspondence between each pixel and its characteristic set of waveforms. In the next subsection, the time sampled versions of these waveforms will be called pixel vectors.

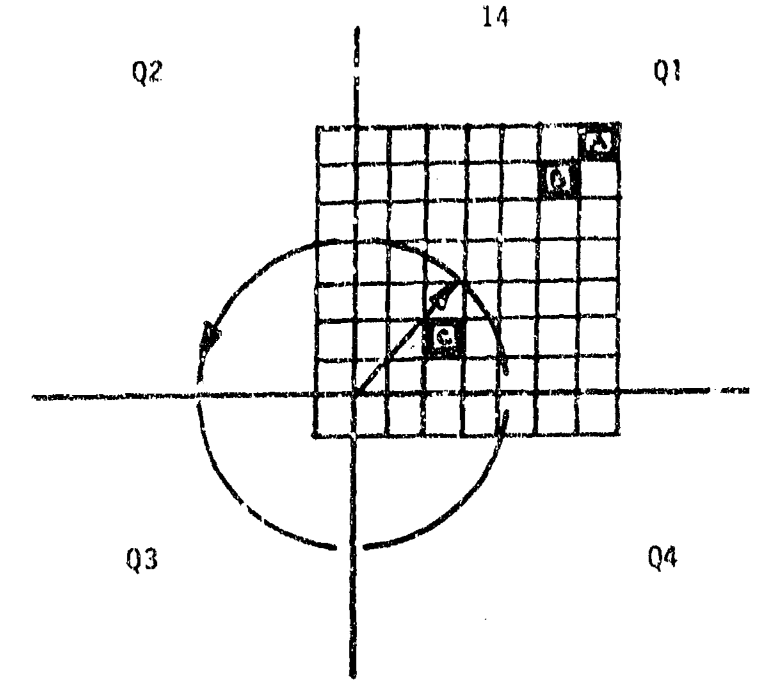
NUTATED QUAD CELL OUTPUT FOR EXTENDED OBJECT





OBJECT SHAPE

FIGURE 2.2



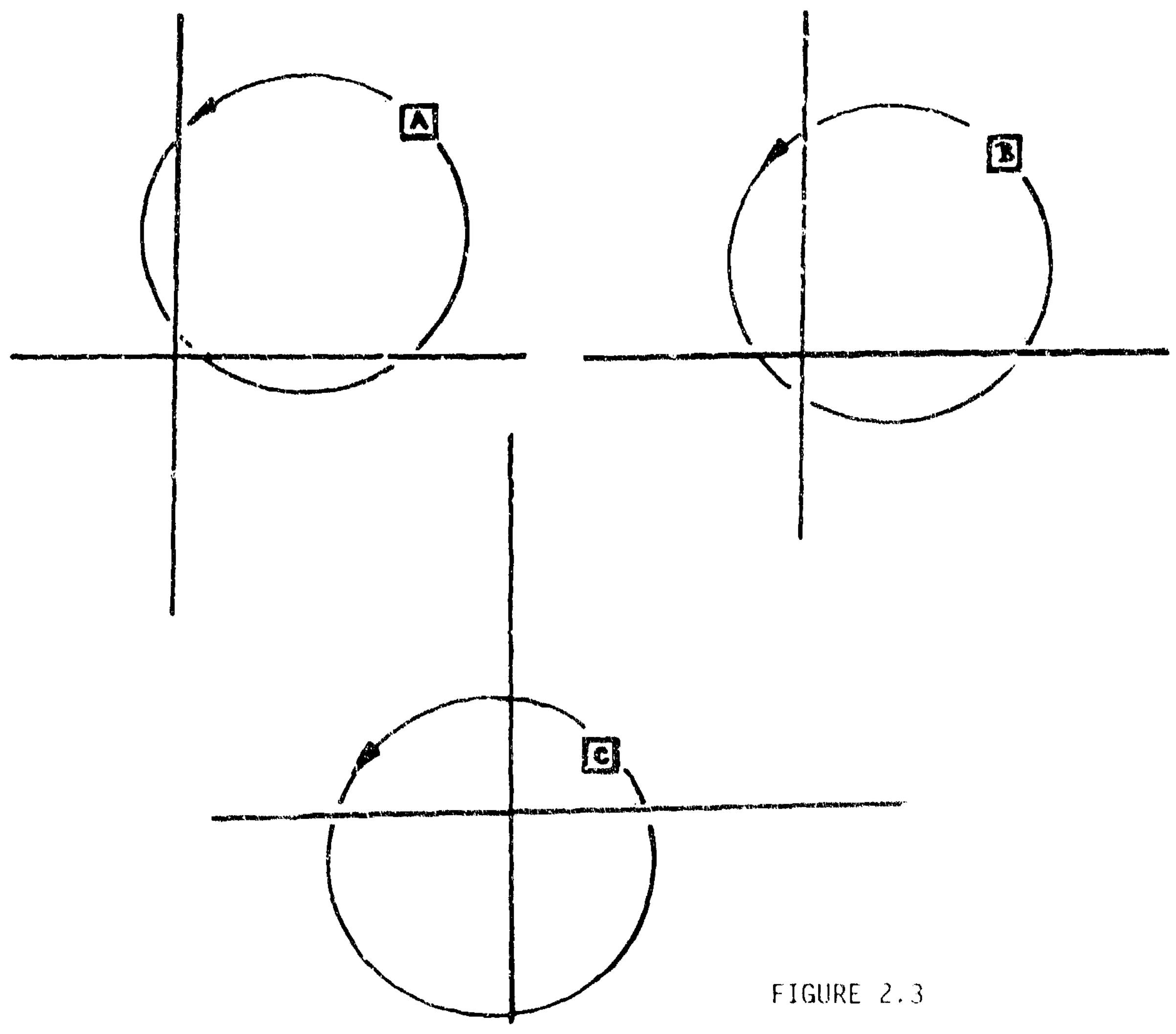


FIGURE 2.3

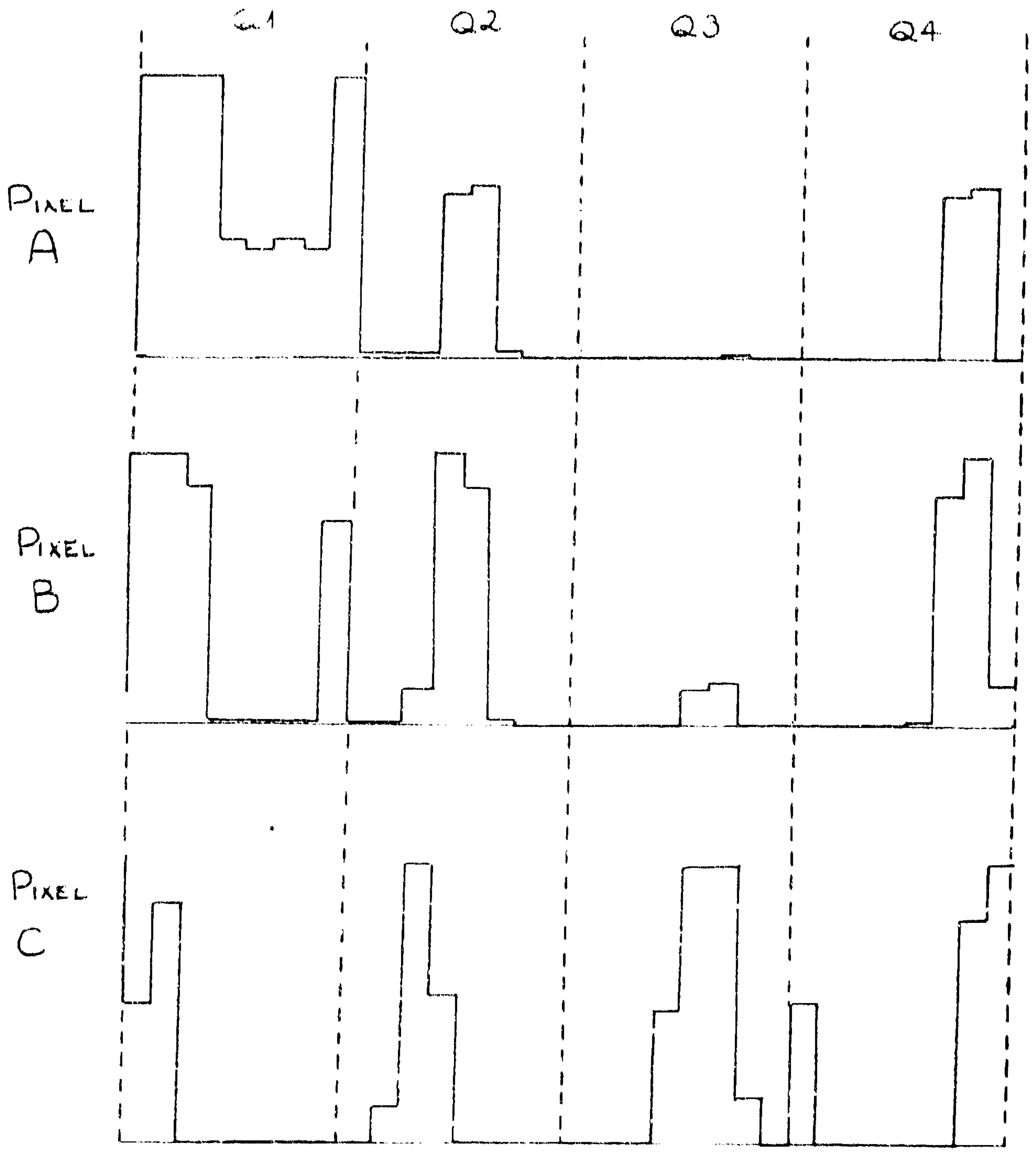


FIGURE 2.4

The process of nutation and quad cell detection can therefore be viewed as a linear transformation between the pixel representation of the image and a pixel vector representation. This concept is illustrated in Figure 2.5. In order that the pixel vector representation be complete, certain sampling constraints must be observed. As will be seen in subsection 2.3, this consideration leads to the use of more complex nutation patterns than the circle (which likewise complicates the tracking process, as seen in Section 3). The image reconstruction problem is then seen as analogous to using an inverse transformation of the nutation process to obtain the original image. In the next section a procedure for finding this transformation is developed.

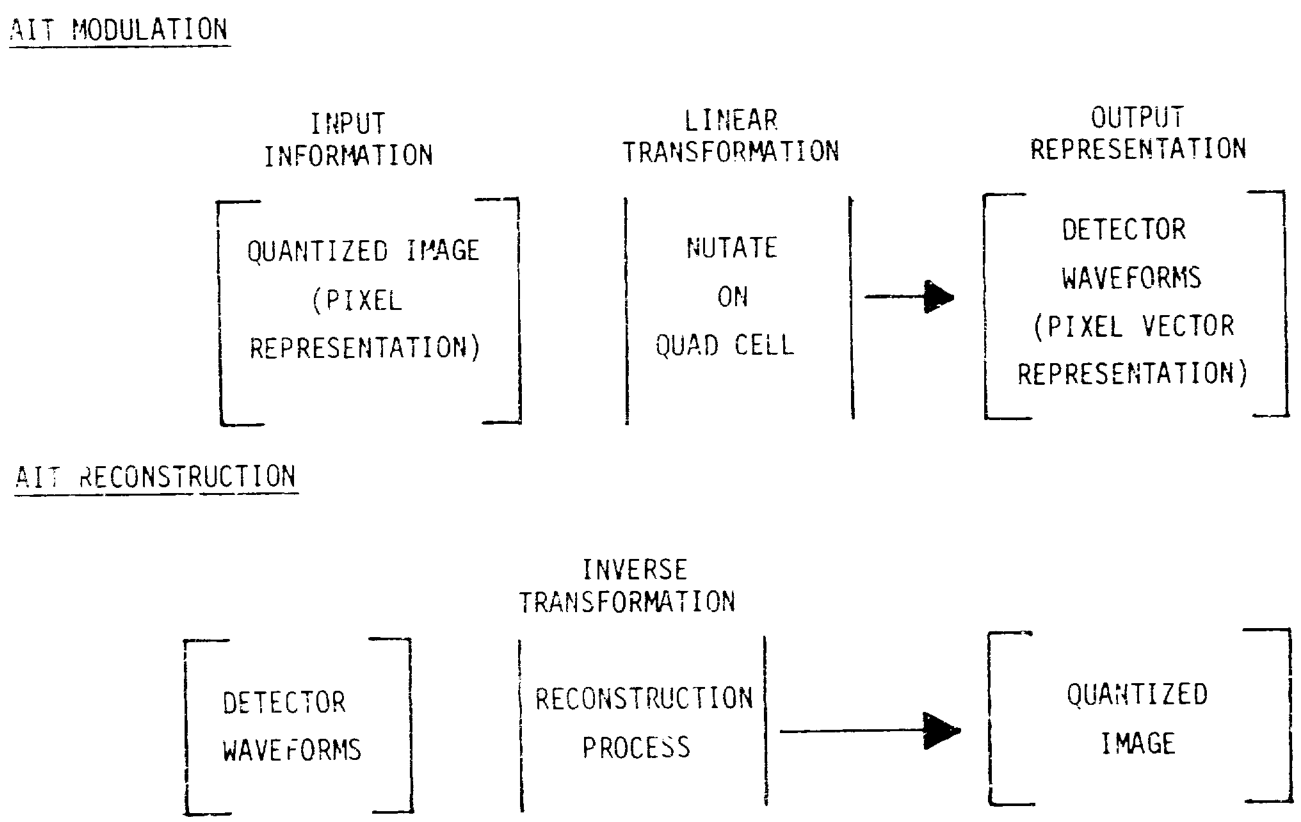

AIT MODULATION
INPUT INFORMATION
QUANTIZED IMAGE (PIXEL REPRESENTATION)
LINEAR TRANSFORMATION
NUTATE ON QUAD CELL
OUTPUT REPRESENTATION
DETECTOR WAVEFORMS (PIXEL VECTOR REPRESENTATION)
AIT RECONSTRUCTION
DETECTOR WAVEFORMS
INVERSE TRANSFORMATION
RECONSTRUCTION PROCESS
QUANTIZED IMAGE

FIGURE 2.5

## 2.3 The AIT Image Reconstruction Algorithm

The AIT imaging algorithm reconstructs the input image by solution of the matrix equation which relates the detector output waveforms to the image intensity distribution for a given nutation pattern. In order to define this equation, we consider an N x N element square pixel field which will contain the image. If the amplitude of the ith pixel is given by the scalar $A_i$, then the entire image can be defined by a vector containing all pixel amplitudes.

$$\vec{A} = \begin{matrix} A_1 \\ \cdot \\ \cdot \\ \cdot \\ A_{N^2} \end{matrix} \tag{2.3}$$

Consider an image in which only the ith pixel is illuminated with unit amplitude, i.e.,

$$\begin{aligned} A_i &= 1 \\ A_{k=i} &= 0 \end{aligned} \tag{2.4}$$

This will result in a specific detector output waveform when nutated on the quad cell. If the four detector outputs are time sampled with $4N^2$ samples taken as a full nutation pattern is scanned, the $16N^2$ detector output samples corresponding to unit illumination of the ith pixel define the vector, called a pixel vector:

$$\vec{P}_i = \begin{matrix} P_{11} \\ . \\ . \\ P_{QM} \\ . \\ . \\ P_{4\,4N^2} \end{matrix} \qquad \begin{matrix} Q = 1 - 4 \quad \text{Quadrant number} \\ \\ M = 1 - 4N^2 \quad \text{Time sample number} \end{matrix} \qquad (2.5)$$

There is a unique pixel vector $\vec{P}$ associated with each pixel in the image field. The detector output vector $\vec{D}$ is defined similarly to $\vec{P}$ but is the response to a general image input $\vec{A}$. The vector $\vec{D}$ can be described as a linear combination of the unit pixel vectors $\vec{P}$, and this relationship can be formalized by arranging the pixel vectors to form a forward matrix $\underline{F}$

$$\underline{F} = \vec{P}_1 \; . \; . \; . \; \vec{P}_{N^2} \qquad (2.6)$$

The relationship between the input image $\vec{A}$ and detector outputs $\vec{D}$ can be stated

$$\underline{F}\,\vec{A} = \vec{D} \qquad (2.7)$$

Since $\vec{D}$ is the measured variable and $\vec{A}$ the desired result (the image), this equation must be solved for $\vec{A}$. This can be done formally by defining a backward matrix $\underline{B}$ which is an inverse of $\underline{F}$.

$$\underline{B}\,\underline{F} = \underline{I} \qquad \underline{I} \text{ is the identity matrix} \qquad (2.8)$$

That gives

$$\underset{\substack{\uparrow \\ \text{matrix}}}{\vec{A}} = \underset{\substack{\uparrow \\ \text{backward} \\ \text{matrix}}}{\underline{B}} \cdot \underset{\substack{\nwarrow \\ \text{detector} \\ \text{vector}}}{\vec{D}} \qquad (2.9)$$

The matrix $\underline{F}$ is non-square (having dimension $N^2 \times 16N^2$ for a nutation pattern with ideal sampling), so that $\underline{B}$ can be any of the possible pseudo-inverses of $\underline{F}$. The pseudo-inverse having minimum variance for noisy input is defined by

$$\underline{B}_{min} = (\underline{F}^T\underline{F})^{-1}\underline{F}^T \tag{2.10}$$

Reconstruction of an image from quadrant detector outputs requires a nutation pattern sufficiently complex to unambiguously sample all pixels in the required reconstruction field. That is, the spatial Nyquist criterion of at least two samples per linear dimension per pixel must be obtained. In terms of AIT, the image must be nutated so that the quad cell axes fall at four distinct sample points in each pixel. For perfectly uniform sampling, this implies $4N^2$ samples per quadrant for a total of $16N^2$ samples needed for an alias-fee reconstruction, where N is the number of pixels across one edge of a square field.

Here the spatial Nyquist criteria of sufficient sampling is specified by the existence of the inverse $(\underline{F}^T\underline{F})^{-1}$.

## 2.4 Properties of the Reconstruction Algorithm

As stated earlier, the back matrix $\underline{B}$ (equation 2.10) will yield the minimum norm estimate of the image from the input data D.(3) The exact relationship between the variance on the reconstructed image pixels and the system noise is derived below.

### 2.4.1 AIT Imaging Noise

The AIT imaging process is a multiplexed detector approach. Although the nutated quad cell detector arrangement yeilds a very efficient centroid tracker, its use as an imager is a compromise from a signal-to-noise standpoint. In the following sections an expression for the noise propagator from detector output to image output is derived.

#### AIT Noise Propagator

We will use the definitions:

$A_i$ = amplitude of ith pixel in input image (ith entry of image vector)

$\Delta A_i$ = noise in ith pixel (zero mean random variable)

$B_{ij}$ = back matrix coefficient for ith pixel and jth detector sample

$D_j$ = jth detector sample (jth entry of detector vector)

$\Delta D_j$ = detector sample noise (zero mean random variable)

We wish to determine the variance squared in the i th pixel after one frame. This will be given by

$$\langle (A_i+\Delta A_i)^2\rangle = \langle (\sum_j B_{ij}(D_j+\Delta D_j))^2\rangle \qquad (2.11)$$

where $\langle\ \rangle$ denotes expected value, and the sum is over the j detector values.

Expanding

$$\langle (A_i)^2 + 2\Delta A_i A_i + (\Delta A_i)^2\rangle = \langle (\sum_j B_{ij}D_j + \sum_j B_{ij}\Delta D_j)^2\rangle$$

$$\langle (A_i)^2\rangle + \underbrace{\langle (2\Delta A_i A_i)\rangle}_{0 \text{ since } \Delta A_i \text{ has zero mean}} + \langle (\Delta A_i)^2\rangle = \langle (\sum_j B_{ij}D_j)^2 +$$

$$2(\sum_j B_{ij}D_j \sum B_{ij}\Delta D_j) + (\sum_j B_{ij}\Delta D_j)^2\rangle$$

$$\langle (\cancel{A_i})^2\rangle + \langle (\Delta A_i)^2\rangle = \langle (\cancel{A_i})^2\rangle + \langle 2(\sum_j B_{ij}D_j \sum B_{ij}\Delta D_j)\rangle +$$

$$\langle (\sum_j B_{ij}\Delta D_j)^2\rangle$$

$$\underbrace{\langle (\Delta A_i)^2\rangle}_{\text{variance squared in } i\text{th pixel}} = \underbrace{\langle 2A_i \sum \Delta D_j\rangle}_{0 \text{ since } \Delta D_j \text{ has zero mean}} + \langle (\sum_j B_{ij}\Delta D_j)^2\rangle$$

$$= \langle (\sum_j B_{ij}\Delta D_j)^2\rangle \qquad (2.12)$$

$$= \langle (\sum_j B_{ij}\Delta D_j)(\sum B_{ij}\Delta D_j)\rangle$$

$$= \langle (\sum_j (B_{ij}\Delta D_j)^2 \rangle \tag{2.13}$$

because terms like $\langle B_{ik}\Delta D_k B_{i\ell}\Delta D_\ell \rangle = 0$

since $\Delta D_j$ is a random variable.

$$= \sum_j \langle (B_{ij}\Delta D_j)^2 \rangle$$

$$\langle (\Delta A_i)^2 \rangle = \sum_j B_{ij}{}^2 \langle \Delta D_j{}^2 \rangle \tag{2.14}$$

We have the result that the variance squared in the i th pixel brightness is given by the variance squared on the detector samples times the sum of the squares of the i th column of the back matrix.

The quantity $\langle D_j{}^2 \rangle$ can be related to the detector noise power by the expression

$$\langle \Delta D_j{}^2 \rangle = \frac{N_{TAIT}}{\Delta T_{FRAME}} (P_{NAIT})^2$$

where $N_{TAIT}$ = number of time steps in nutation pattern; minimum 4N, where N is total # of pixels

$\Delta T_{FRAME}$ = duration of nutation pattern (frame time)

$P_{NAIT}$ = noise power for detectors in $(Hz)^{-1/2}$ (detector must cover full 8×8 F.O.V.)

∴ for AIT

$$\langle (\Delta A_i)^2 \rangle_{AIT} = \sum_j B_{ij}{}^2 \frac{N_{TAIT}}{\Delta T_{FRAME}} (P_{NAIT})^2 \tag{2.15}$$

## 2.4.2 AIT image conditioning characteristics

The formulation of the reconstruction algorithm allows a certain amount of image conditioning to be included in the back matrix $\underline{B}$, which can have significant engineering implications. Specifically if the forward matrix $\underline{F}$ is generated by a particular hardware AIT system by sequentially presenting the system with point source illumination in each of the pixels in the input representation field and recording the detector outputs, $\underline{F}$ will contain all the effects of the system optical transfer function. The back matrix $\underline{B}$ will include the inverse of the OTF. As a result, resolution loss due to optical aberration, image distortion, and other image defects can be removed in the reconstruction process. The extent to which this can be done depends on the characteristics of the system OTF; significant amounts of deblurring have been achieved using similar techniques.(4) This image conditioning property arises as a desirable side benefit of this reconstruction approach.

## 3.0 THEORETICAL FORMULATION: TRACKING

### 3.1 Algorithm Constraints

The twin needs of imaging and tracking act as competitive constraints on the nutation pattern used to scan the object. The spatial Nyquist criteria (Section 2.3) dictate the sampling density needed to provide imagery at a specific resolution. The desire for fast tracking, using an $I^3$ Sensor-like tracker algorithm, requires a sampling sequence which circles the quad cell origin many times for each complete sample set in order to get many centroid estimates per image estimate. In addition, engineering limitations on the nutation device itself, which will be discussed in Section 7, have led to use of a specific class of compound Lissajous figures for nutation patterns. The image field is deflected according to the following set of parametric equations:

$$X_{nut} = Dmax \left\{(\sin 2\pi f_n t)\ (\cos 2\pi (f_n/s)t)\right\} \qquad (3.1)$$

$$Y_{nut} = Dmax \left\{(\cos 2\pi f_n t)\ (\sin 2\pi (f_n/s)t)\right\}$$

Dmax = pattern size,

$f_n$ = nutation frequency,

S = number of spirals per pattern.

This pattern, referred to as the collapsing ellipse pattern, can be seen as composed of a circular nutation pattern of nutation frequency $f_n$, multiplied by an amplitude modulation function at frequency $f_n/s$. The sampling sequence therefore makes S circuits about the quad cell center for every complete pattern. An example of this pattern, which was used as the benchmark throughout the program, is shown in Figure 3.1. The pattern was designed to cover an 8 x 8 pixel imaging field. The parameter S = 15, and there are 31 sample points taken every spiral, for a total of 31 x 15 = 465 samples per image.

Given these basic nutation characteristics, a tracking algorithm can be defined.

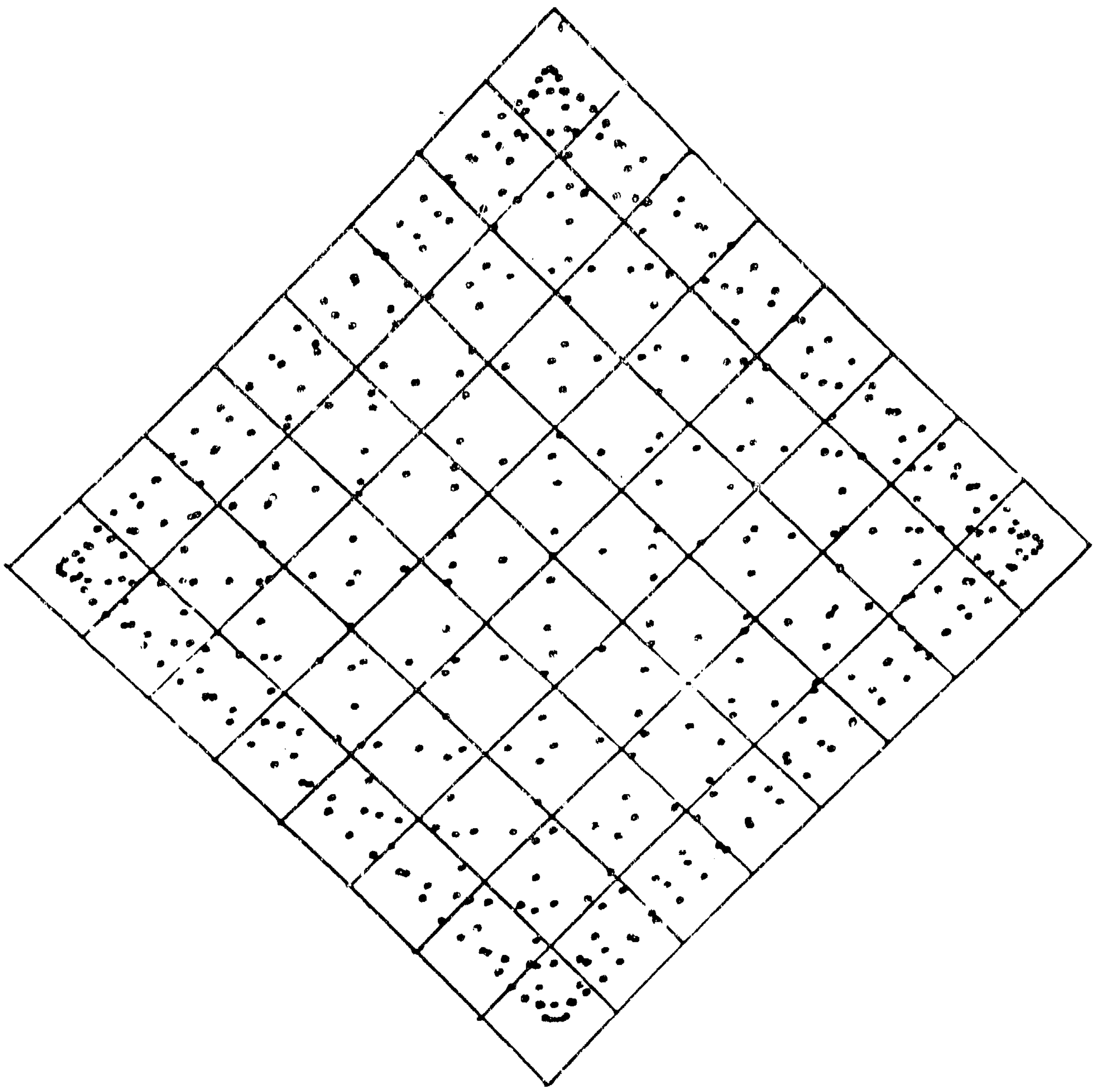


FIGURE 3.1

## 3.2 The SCS Algorithm

The SCS algorithm can be defined by reference to Figure 3.2. A focal spot corresponding to the target image is nutated on a quad cell. The quadrants are labeled $Q_1$ - $Q_4$. Four intervals in the nutation cycle are also defined, as periods $T_1$ - $T_4$, or nutation angular intervals $\theta_1$ - $\theta_4$, which are equivalent since with circular nutation the spot velocity is constant. Measurements of spot centroid displacement are made at different times for the two axes; x-displacement is measured during periods $T_1$ and $T_3$, while y is measured during $T_2$ and $T_4$. The algorithm defines the spot position, and therefore input tilt, by a numerator proportional to displacement and brightness, and a denominator proportional to brightness only:

$$X_{NUM} = (Q_1(T_2)-Q_2(T_2)) - (Q_4(T_2)-Q_3(T_2)) + (Q_4(T_4)-(Q_3(T_4)) - (Q_1(T_4)-Q_2(T_4)) \qquad (3.2)$$

$$X_{DEN} = (Q_1(T_2)+Q_2(T_2)) - (Q_4(T_2)+Q_3(T_2)) + (Q_4(T_4)+(Q_3(T_4)) + (Q_1(T_4)+Q_2(T_4)) \qquad (3.3)$$

$$X = \frac{X_{NUM}}{X_{DEN}} \qquad (3.4)$$

The motivation for this definition can be seen by separately examining the effects of each term. The differences $Q_1(T_2)-Q_2(T_2)$ and $Q_4(T_4)-Q_3(T_4)$ correspond to measuring the intensity imbalance between right and left half-planes, during those times when the spot is expected in those quadrants, which occurs due to spot displacement. The imbalance occurs because a shift of the spot appears as a shift in the center of the nutation circle, as shown in Figure 3.3. The imbalance between the signals integrated from each quadrant can be interpreted as proportional to the differences in arc lengths bounded by the integration intervals and quad cell boundaries, as shown in the figure.

The other numerator terms $Q_4(T_2)-Q_3(T_2)$ and $Q_1(T_4)-Q_2(T_4)$ are the power differences seen when the focal spot is <u>not</u> present in the respective quadrants. For an ideal, sharply bounded spot, the second set of terms should represent differences only in background radiation or detector responses, and serve to cancel these contributions from the first set, so that the numerator becomes sensitive to the spot only. For spots whose radii exceed the size of the nutation radius, or for less well bounded spots (such as the Airy pattern or Gaussian distributions commonly encountered) the spot intensity is never wholly absent from a quadrant even though nutation may have moved the spot farthest away from that quadrant during a cycle; in these cases spot power as well as background power is cancelled from the numerator, and modulation efficiency is reduced. The relationship between object size and nutation radius is therefore an important consideration.

For objects smaller than the nutation radius, the scale of the displacement is given by the nutation radius, that is, the output of the algorithm gives displacements as fractions of the nutation radius. The response of the algorithm as a function of nutation radius is summarized in Figure 3.4. At small radii, the sensitivity is determined by the spot size, while at large radii the nutation radius sets the sensitivity.

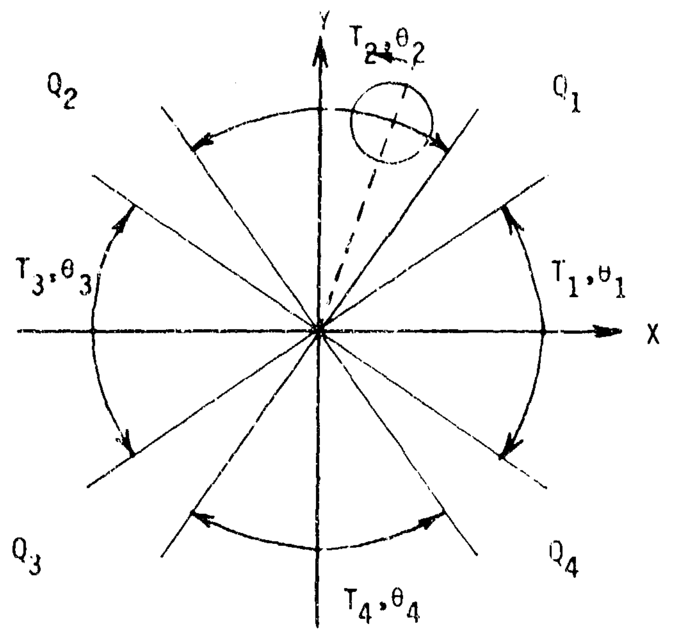


FIGURE 3.2

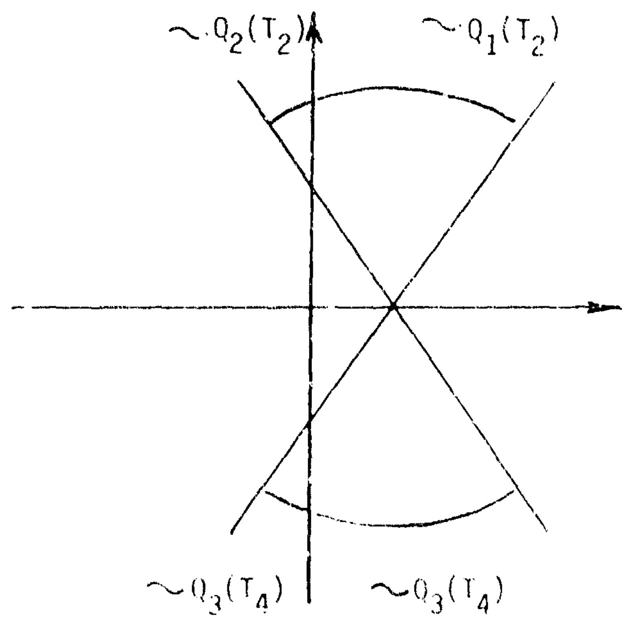


FIGURE 3.3

The function of the denominator is to normalize the output to spot brightness. It can be seen as a sum of left and right half-plane terms, rather than a difference as in the numerator. Background cancelling terms are still present.

The algorithm for y-axis information is similarly defined, using the time intervals $T_1$ and $T_4$. Note that y-axis information is obtained $90^o$ out of phase with x-axis information during the cycle.

It is important to emphasize that the background-cancelling terms make the SCS algorithm an AC algorithm. The position information is impressed on a carrier at the nutation rate. Out of band noise, in particular 1/f noise, is rejected by the same cancelling action as removes background illumination.

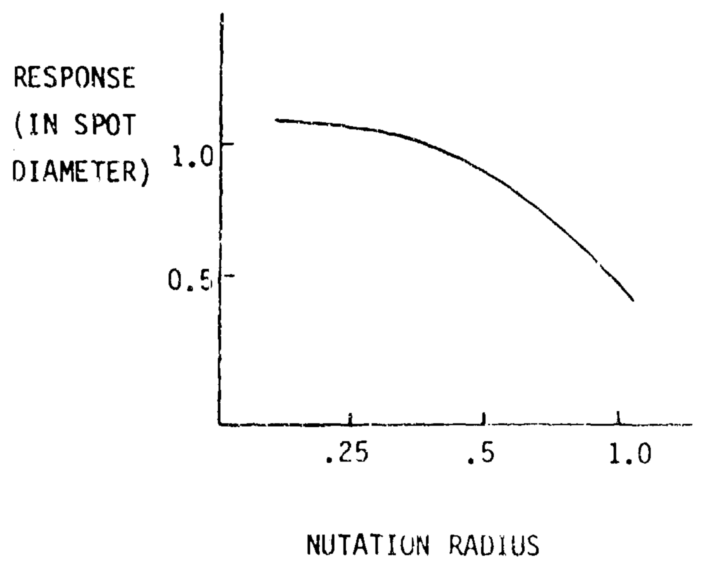


FIGURE 3.4

## 3.3 The AIT Tracking Algorithm

Significant modifications must be made to the SCS algorithm to accommodate the AIT nutation pattern. Some performance loss might be expected, but the following features should be retained:

1) AC operation. The continued recognition that many of the AIT applications will be in infrared systems means that AC demodulation must be implemented to avoid 1/f noise and uniform background illumination problems.

2) Bandwidth. A closed figure is described once every 15 times the spot orbits the center; imaging is updated every time a full pattern is completed. This will be referred to as a major cycle, as shown in Figure 3.5. The separate traversals will be denoted minor cycles; one example is shown in Figure 3.6. In order to retain the AIT feature of tracking at greater rates than imaging, tracker data should be obtained at the minor cycle rate, which is 15 times that of the major cycle, or imaging rate. This complicates the problem further, since each minor cycle is different from the others, requiring a different algorithm form for each.

3) Well-defined scale factor. In the SCS algorithm, the scale factor is defined by the nutation radius as long as the object size is smaller than that radius. In the AIT nutation pattern, it is clear that there exists no single nutation radius, so that the scaling of the output becomes less obvious. Since varying size objects will be observed, the algorithm must be adaptable to varying object extent.

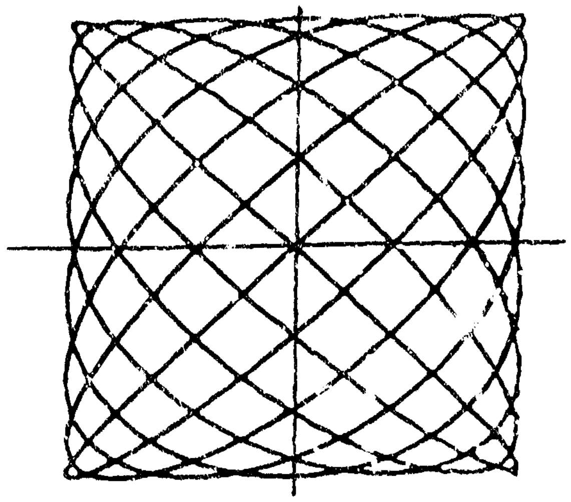


FIGURE 3.5. AIT MAJOR CYCLE

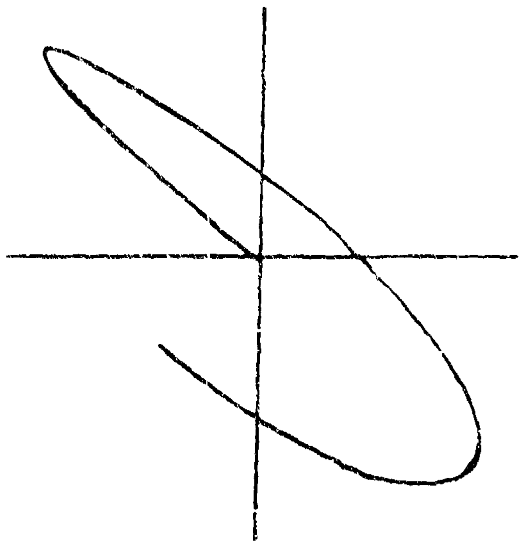


FIGURE 3.6. AIT MINOR CYCLE

Using these criteris, we now outline a general prescription for an AIT tracker algorithm, which uses a form analogous to eq. (3.2) - (3.4):

$$X_{num}(M) = \sum_{N_2(M)} (Q_1(N_2) - Q_2(N_2)) - \sum_{N_2(M)} (Q_4(N_2) - Q_3(N_2)) +$$

$$\sum_{N_4(M)} ((Q_4(N_4) - Q_3(N_4)) - \sum_{N_4(M)} (Q_1(N_4) - Q_2(N_4)) \quad (3.5)$$

$$X_{den}(M) = \sum_{N_2(M)} (Q_1(N_2) + Q_2(N_2)) - \sum_{N_2(M)} (Q_4(N_2) + Q_3(N_2)) +$$

$$\sum_{N_4(M)} (Q_4(N_4) + Q_3(N_4)) - \sum_{N_4(M)} (Q_1(N_4) + Q_2(N_4)) \quad (3.6)$$

$$X(M) = S_X(M)\frac{X_{num}(M)}{X_{den}(M)} + X_{off}(M) \quad (3.7)$$

Here the quadrant outputs are referenced to a sample number N of a minor cycle M, where for the current AIT $N \leq 31$, $M \leq 15$, the separate sets of samples $N_1$ - $N_4$ are analogous to the intervals $T_1$ - $T_4$ in the SCS algorithm, and define which samples are used to form an interval. This procedure will be discussed in detail below. The term $X_{off}(M)$ is an offset correction term that accounts for residual asymmetry in the sample set, which differs for different minor cycles. The factor S(M) is a scale correction factor, which defines a nominal or average nutation radius for the cycle.

Consider the example of Figure 3.7. A minor cycle is shown (actually the most difficult of the cycle shapes to treat), with its 31 sample segments marked. The two circles indicate two example object sizes of radius 1 pixel (2 pixel diameter) and 2

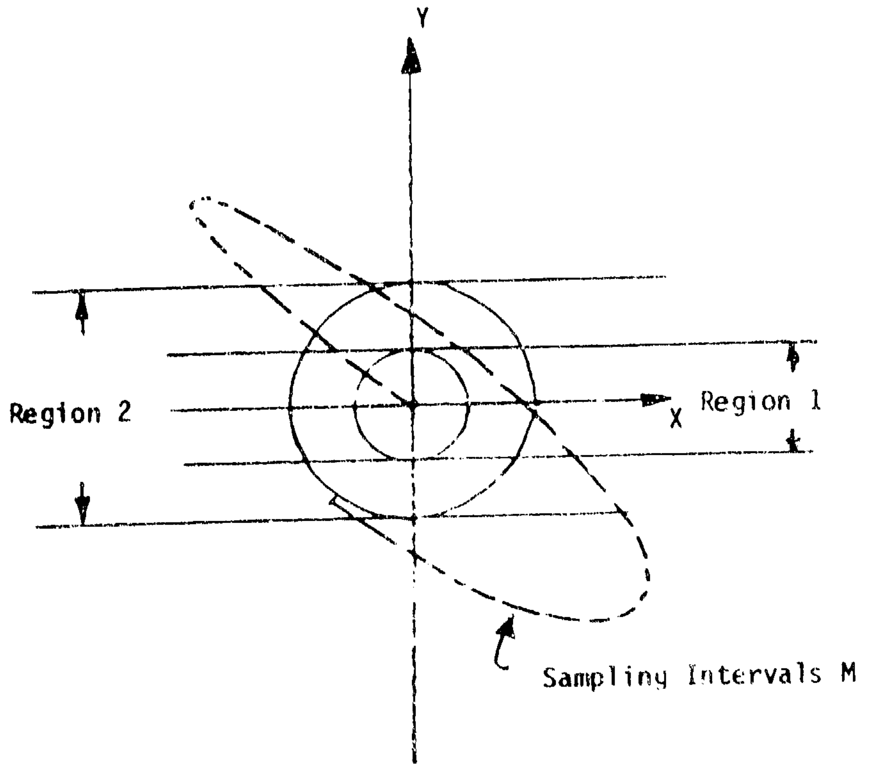


FIGURE 3.7

pixels (4 pixel diameter). To measure an X-displacement using an SCS-type algorithm, the object must be entirely above or below the x-axis. For Y-measurements, the object must be completely to the left or right of the y-axis. This insures that the measurement is independent of object size. Based on this notion, all samples outside Region 1 can be included for the small obejct, and those outside Region 2 can be used with the large object.

A second constraint is that a sampling interval must extend at least an object width to either side of the axis which it crosses during the measurement interval. The intervals ultimately defined by these constraints are shown in Figures 3.8 and 3.9 for the two and four pixel diameter objects respectively.

Note that for the four pixel spot, x-axis information cannot be obtained in this minor cycle, since there are no samples in Quadrant 1. Y-information for this large object would have to be obtained from other minor cycles with more amenable shapes.

The response of the algorithm for any chosen minor cycle and object size can be calculated for a point-object, essentially by measuring the x- or y-axis components of the included arc lengths in the measurements. By obtaining values of X(M) for a zero-offset spot and a unit offset spot (for a given M), the constants S(M) and $X_{off}$(M) can be obtained from equation (3.7).

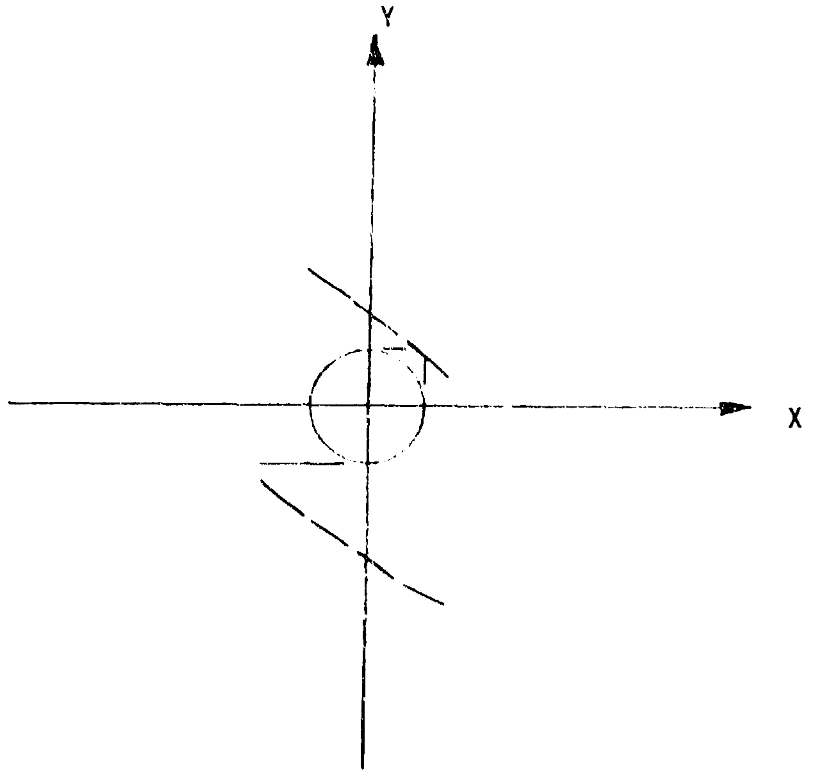


FIGURE 3.8. Usable arc lengths for 2 pixel diameter object.

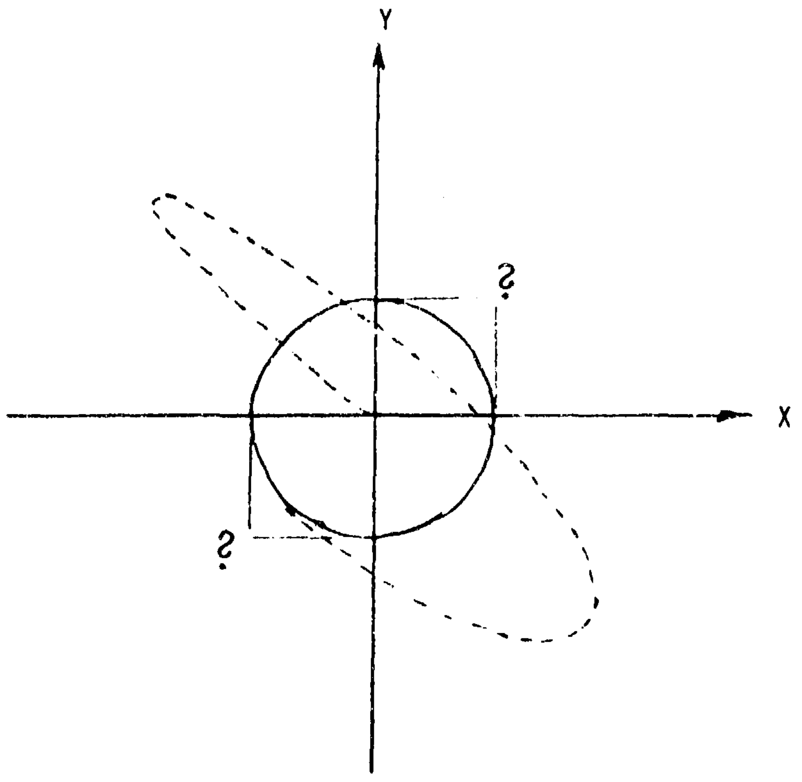


FIGURE 3.9. Object size (4 pixel) exceeds available arc lengths.

## 3.4 AIT Tracker S/N

The expression for angular measurement variance, eq. (2.1), of the $I^3$ Sensor can be modified to provide an estimate of the AIT tracker S/N performance. Two observations must be made. One, eq. (2.1) is true for a system operating at optimal nutation radius, that is

$$r_N = \frac{\lambda}{D} \tag{3.8}$$

where $r_N$ is the nutation radius, $\lambda$ the received wavelength, and D the subaperture diameter. The ratio $\lambda/D$ is the angular resolution of the aperture, so it is seen that best performance occurs with the angular scan radius equal to the pupil angular resolution. In general, for nutation radii larger than $\lambda/D$, the variance is given by:

$$\sigma^2 = \frac{2}{SNR} (R_N)^2 \tag{3.9}$$

where $R_N$ is the (not necessarily optimal) angular nutation radius. For AIT, $R_N$ can be replaced by a mean or normalized nutation radius corresponding to an average over the arc used in each minor cycle. For AIT, $R_N$ will nearly always be larger than optimal, to maintain object size insensitivity.

In addition to the variance increase encountered due to nutation size increase, there is also a duty cycle reduction that results from use of only part of the total available arc length; the loss to performance is not exactly equal to the fractional loss of arc length since the samples taken at different nutation angles contribute differently to the measurement. As a worst case, however, one can assume that the variance is proportionate to the fraction of a nutation cycle used:

$$\beta_{i,j} = \frac{\theta_N}{\pi} \tag{3.10}$$

where $\beta_{i,j}$ refers to the i th axis (x or y) of the j th minor cycle, and $\theta_N$ is the total nutation included by the minor cycle.

An upper bound on the AIT tracker variance can therefore be given as

$$\sigma^2_{i,j} \leq \frac{2}{SNR} R_N^2 \beta_{i,j} \tag{3.11}$$

## 4.0 SOFTWARE SIMULATION AND PERFORMANCE CHARACTERIZATION: IMAGING

All of the image reconstruction algorithms were characterized using an extensive set of simulation codes run on the AOA Data General computer system. This section discusses the structure and function of the major simulation packages, the procedures involved in constructing the back matrix, and the results obtained using the process to reconstruct images from simulated data sets.

### 4.1 Computer System

The simulations were run on the Air Force-owned Data General Nova 3/12 system located at AOA. The configuration of this micro-computer system is diagrammed in Figure 4.1. The computer is equipped with 64K words of mapped memory, hardware floating point capability and a ULM I/O port for communication with extra external devices (such as the printer and the PMP) under RS-232 protocol. The system supports two interactive users in foreground/background mode. A floppy disk and a cartridge disc drive (5 MByte fixed, 5 removable) are for mass storage. Hard copy of text is output through a DG/Dasher printing terminal.

Two peripheral systems communicate with the Nova through its backplane, the DG/DAC system and the color display. The DG/DAC laboratory interface includes 4 A/D channels, D/A channels, and 16 bits each of latched TTL inputs and outputs. This system enables computer access to experimental hardware for control and data collection purposes.

This interface was also used to access the AIT interface electronic system, which contains the nutation drive generator and a switched integrator front end for data sampling of the nutation waveforms.

The Genisco Programmable Graphics Processor and associated hardware provide high-speed, medium resolution color graphics capability for pseudo-color image output. This particular system was chosen for its high data transfer rate and color-fill speed, which could accommodate the anticipated maximum AIT image rate.

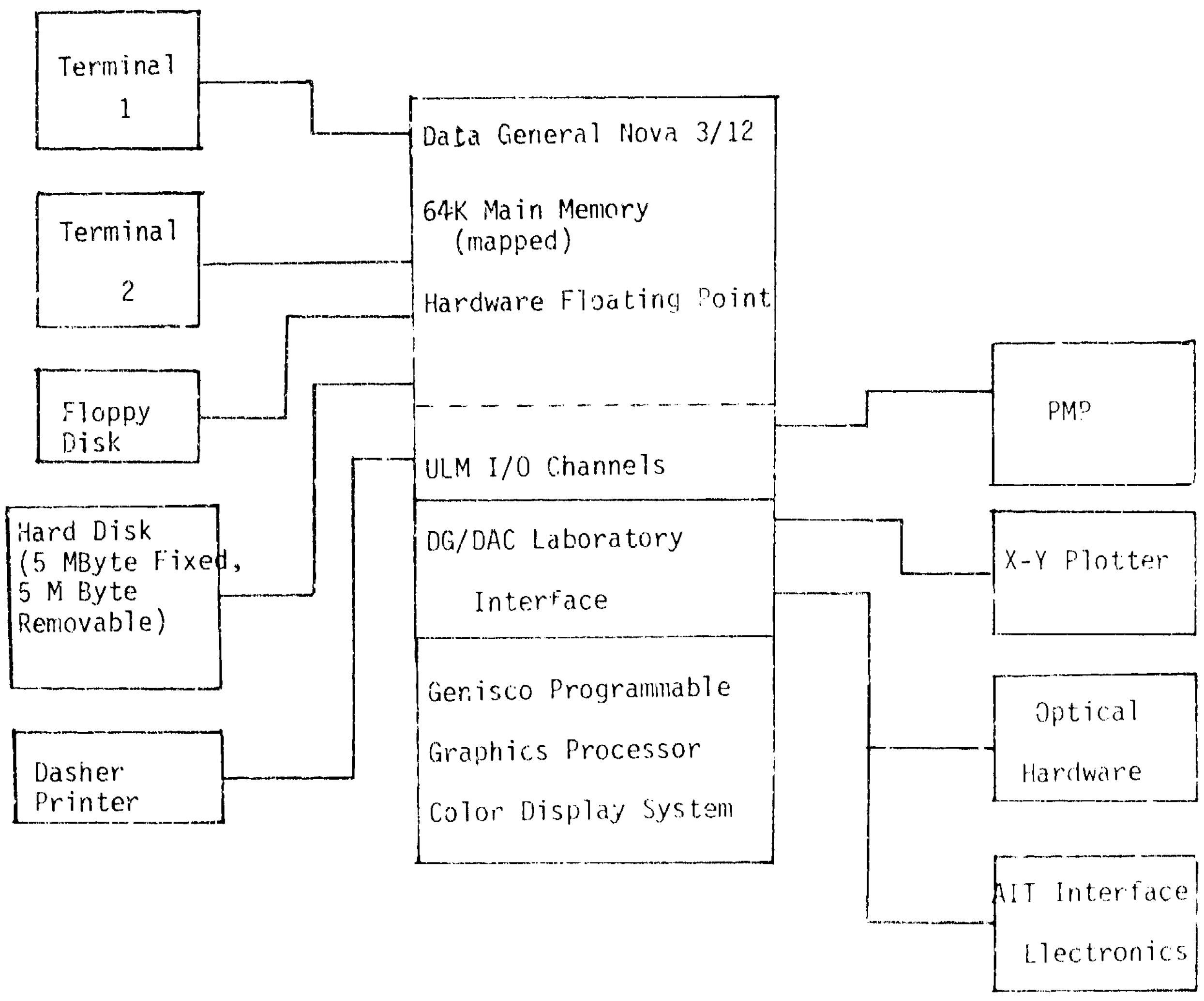


FIGURE 4.1. AOA COMPUTER SYSTEM

## 4.2 Simulation Structure

To provide unambiguous characterization of algorithm performance and retain the operational flexibility needed to conduct algorithm development, the program structure shown in Figure 4.2 was adopted. Depicted are three system segments, two of which are software constructs.

Complete simulation of a proposed AIT configuration first requires a means for generation of the nutation waveforms associated with the desired input test pattern. This function is performed by program SIMDSK. Originally written to simulate $I^3$ Sensor outputs under the $I^3$ Sensor Study program, SIMDSK was expanded to accommodate the more complex nutation patterns used in AIT. SIMDSK is designed to parallel as closely as possible the physical process as done in the experimental set-up. The program accepts a test image defined as a 20 x 20 array of intensity values. This input file is then numerically superimposed on a 40 x 40 element array which defines the quad cell. The quad cell routine can include the effects of nonuniform responsivity and finite gaps between detector elements. The 2-D integral of the power in each detector is calculated and output.

The displacement between the image and quad cell coordinate systems can be set by the operator (as an input tip and tilt) and by another file containing the description of the nutation pattern. The detector output calculation is repeated for each of the spot positions defined in the nutation file. The result is a vector of detector values given as a function of nutation, i.e. a quad cell waveform set.

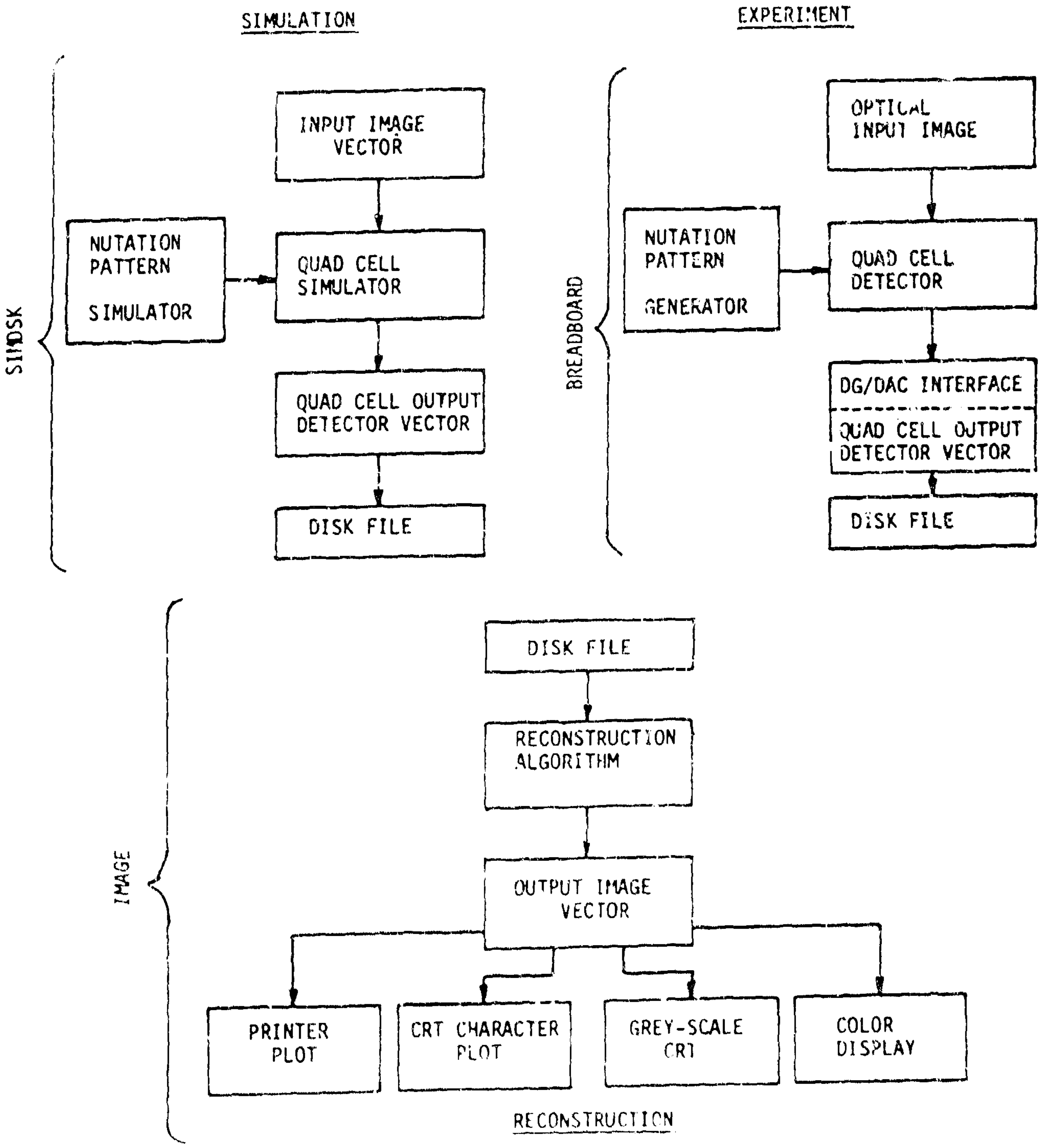

SIMULATION
EXPERIMENT
SIMDSK
INPUT IMAGE VECTOR
NUTATION PATTERN SIMULATOR
QUAD CELL SIMULATOR
QUAD CELL OUTPUT DETECTOR VECTOR
DISK FILE
BREADBOARD
OPTICAL INPUT IMAGE
NUTATION PATTERN GENERATOR
QUAD CELL DETECTOR
DG/DAC INTERFACE
QUAD CELL OUTPUT DETECTOR VECTOR
DISK FILE
IMAGE
DISK FILE
RECONSTRUCTION ALGORITHM
OUTPUT IMAGE VECTOR
PRINTER PLOT
CRT CHARACTER PLOT
GREY-SCALE CRT
COLOR DISPLAY
RECONSTRUCTION

FIGURE 4.2

The simulated quad cell outputs are stored in a disk file for later access. Generation of a detector waveform can take up to an hour, depending on the resolution used in the calculation.

For reconstruction of the nutated image, the program IMAGE is used. The input to image is the disk file containing a nutation waveform. A reconstruction algorithm is then called, which is selectable by the user. Normally this would involve data conditioning steps and then multiplication of the detector vector by the back matrix. The output image vector can then be displayed by a variety of peripheral devices, such as a printer plot, grey-scale oscilloscope display, or the Genisco color display.

## 4.3 Generation of the Back Matrix

The computational magnitude of the pseudo-inverse generation step is considerable, and at this time the problem will be scaled by introducing the specific parameters used in the AIT program, as listed in Table 4.1.

| Parameter | Size | Quantity | Dimension |
|---|---|---|---|
| Pixel field size | 8 x 8 | $\vec{A}$ | 64 pixels |
| Number of minor cycles per pattern | 15 | S | |
| Number of samples per minor cycle | 31 | M | |
| Number of samples per pattern | 4 x 15 x 31 | $\vec{D}$ | 1860 |
| Size of forward matrix | 64 x 1860 | $\underline{F}$ | 119,040 coefficient |
| Size of normal matrix | 64 x 64 | $(\underline{F}^T\underline{F})$ | 4096 entries |
| Size of back matrix | 1860 x 64 | $\underline{B}$ | 119,040 |

TABLE 4.1. PARAMETERS OF AIT RECONSTRUCTION PROBLEM

Because of the limited computation speed and particularly the limited memory size of the Nova computer, the problem had to be broken into many sections. Two kinds of routines were designed. One type is a set of computational routines for performing matrix algebraic operations such as matrix multiplication and inversion. The other is a set of disk file handling programs for packing and manipulating the large matrices and minimizing time consuming disk accesses by careful buffering. Particularly important is the design of the transpose routine, which can take up to six hours to transpose a forward matrix if no storage symmetries are exploited.

The computational steps used to generate a back matrix $\underline{B}$ are listed in Figure 4.3. Each block shown contains the name of the program step, the name of the main subroutine called (in parenthesis) and the name of the output file (following the arrow). The output file(s) for each step form inputs for the subsequent steps.

The first version of this code required nearly twelve hours of computer time to generate $\underline{B}$ but once the reconstruction parameters were fixed and the buffering optimized, the back matrix generation time was reduced to about an hour.

During development of these codes, several important numerical constraints were discovered, pertaining to calculational accuracy needed at various stages in the procedure:

1. The determinants of the $(\underline{F}^T\underline{F})$ matrices are very large (of order $10^{64}$), and can achieve values during the inversion process which overflow the machine if attention is not paid to this problem.

2. The three computation intensive steps, in which $\underline{F}^T\underline{F}$, $(\underline{F}^T\underline{F})^{-1}$, and $(\underline{F}^T\underline{F})^{-1}\underline{F}^T$ are formed, must be done using double precision arithmetic for meaningful results to be obtained (on the 16 bit Nova 3).

3. The input data, i.e. the nutation pattern NUTXY and the forward matrix $\underline{F}$ need <u>not</u> have double precision accuracy in order to obtain a well-defined solution. This has important implications for the hardware impelmentation, suggesting that there will be no unanticipated restrictions on the accuracy of the nutator or data collection process.

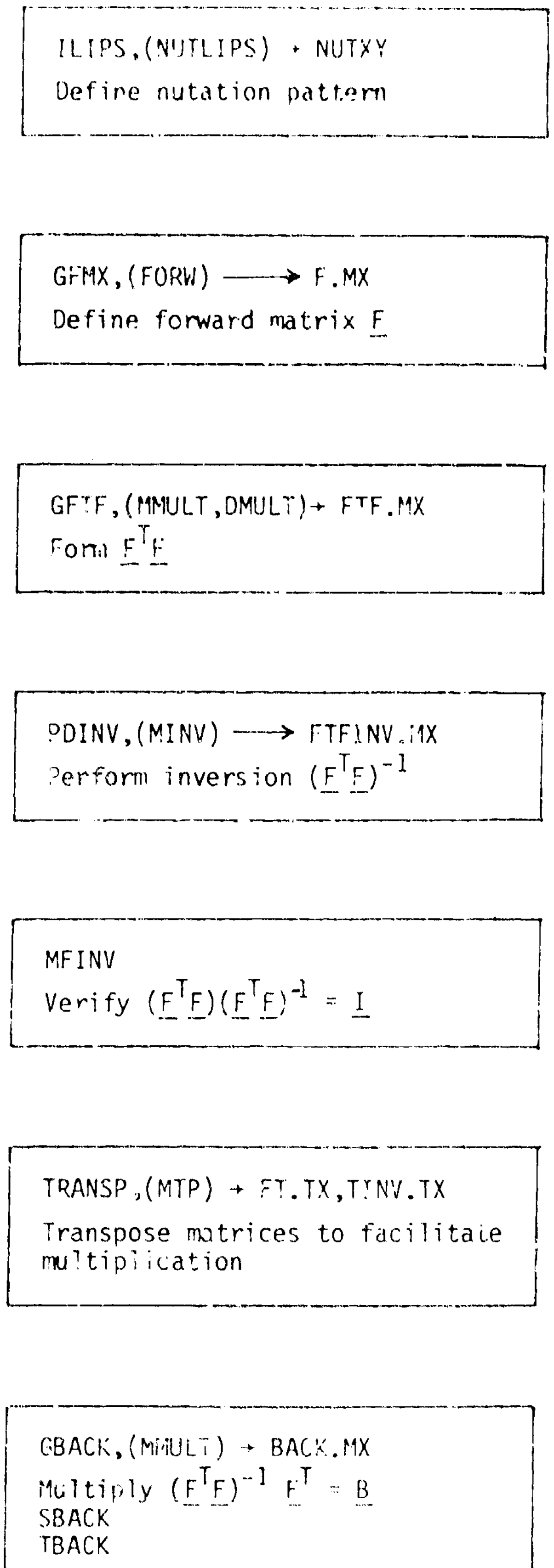

ILIPS,(NUTLIPS) → NUTXY
Define nutation pattern
GFMX,(FORW) → F.MX
Define forward matrix $\underline{F}$
GFTF,(MMULT,DMULT) → FTF.MX
Form $\underline{F}^T\underline{F}$
PDINV,(MINV) → FTFINV.MX
Perform inversion $(\underline{F}^T\underline{F})^{-1}$
MFINV
Verify $(\underline{F}^T\underline{F})(\underline{F}^T\underline{F})^{-1} = \underline{I}$
TRANSP,(MTP) → FT.TX,TINV.TX
Transpose matrices to facilitate multiplication
GBACK,(MMULT) → BACK.MX
Multiply $(\underline{F}^T\underline{F})^{-1}\ \underline{F}^T = \underline{B}$
SBACK
TBACK

FIGURE 4.3

## 4.4 Image Quality

The expected quality of the images produces by this system can be deduced by examination of the identity matrix formed by the product $\underline{B}\ \underline{F} = \underline{I}$. We ask the question: how well are single pixels reproduced by the imaging process?

The first row of the identity generated by the minimum variance $\underline{B}\ \underline{F}$ product is shown in Figure 4.4. The B-matrix was obtained using the procedure described in the previous section, with the F-matrix generated by the 15/31 collapsing ellipse pattern. The row contains the coupling coefficients for each of 64 input pixels into output pixel number one. The coupling is truly excellent. Input pixel one is transferred to output pixel one with unit amplitude to the accuracy of the representation. Contribution from other pixels to output pixel one are down by at least 12 orders of magnitude. Similar accuracy is obtained for the 63 other pixels.

As a point of comparison, the first row of the $\underline{B}\ \underline{F}$ product resulting from a back matrix generated by an iterative technique previously developed during the program (see Phase I Interim Report) is shown in Figure 4.5. The coupling to the correct pixel is only about 93%, while significant transfer occurs from other input pixels (note especially pixel 10 - row 2 column 2 - which has a coefficient of -.126. This is a very significant pixel-pixel interaction). These results were still subjectively acceptable, as seen in the PHASE I interim report results.

```
 1
 1.00      -0.4160-13-0.1440-12-0.1770-13-0.7870-13-0.5430-13-0.5400-13-0.3740-13
-0.1320-12-0.2380-13-0.1170-12-0.4590-13-0.6310-14-0.4120-13-0.5550-13-0.5070-13
-0.7890-13 0.3840-14-0.2980-13-0.4180-13-0.3420-13-0.4270-13-0.1570-13-0.4800-13
-0.4710-13-0.6680-13-0.6520-14-0.5370-13-0.2610-13-0.3380-13-0.6900-13-0.3470-13
-0.8960-13-0.5290-13 0.2280-13-0.5990-13-0.8200-13-0.5670-13-0.4820-13-0.3800-13
-0.3760-13-0.3570-13-0.5320-13-0.7790-14-0.2100-13-0.5870-13-0.3970-13-0.3180-13
-0.5830-14-0.6030-13-0.2960-13-0.7330-13-0.6140-13-0.4050-13-0.2790-13-0.2230-13
-0.3630-13-0.5480-13-0.4390-13-0.3230-13-0 3040-13-0.3740-13-0.2150-13-0.1740-13
```

FIGURE 4.4

```
 1
 0.925      0.374E -1-0.255E -1 0.253E -2 0.304E -3 0.228E -3-0.401E -4-0.669E -3
 0.975E -1-0.126       0.304E -1 0.328E -3-0.198E -2-0.131E -3 0.132E -3 0.954E -3
-0.254E -1 0.304E -1-0.209E -2-0.546E -2 0.238E -2 0.729E -4-0.231E -3-0.435E -3
 0.243E -2 0.462E -3-0.544E -2 0.332E -2-0.507E -3-0.497E -3 0.288E -3 0.131E -3
 0.343E -3-0.136E -2 0.232E -2-0.477E -3-0.601E -3 0.541E -3-0.524E -4-0.923E -4
 0.209E -3-0.134E -3 0.111E -3-0.517E -3 0.539E -3-0.145E -3-0.136E -3 0.832E -4
-0.147E -4 0.729E -4-0.195E -3 0.322E -3-0.461E -4-0.140E -3 0.522E -4 0.473E -4
-0.622E -3 0.375E -3-0.456E -3 0.131E -3-0.506E -4 0.232E -4 0.471E -4-0.889E -4
```

FIGURE 4.5

## 4.5 Evaluation of Image Noise

Equation 2.15 in Section 2.4.1 defined the equation for the noise associated with the AIT reconstruction process:

$$\langle (\Delta A_i)^2 \rangle_{AIT} = \sum_j B_{ij}^2 \frac{N_{TAIT}}{\Delta T_{FRAME}} (P_{NAIT})^2 \qquad (2.15)$$

Once a back matrix has been defined, the quantity $\sum_j B_{ij}^2$ can be evaluated. For comparison purposes, expressions for the pixel noise associated with other imaging techniques have been derived:

For a staring array FLIR

$\sum_j B_{ij}^2 = 1$ , since only one coefficient $B_{ij} = 1$

$N_T = 1$ (no scanning)

$$\langle (\Delta A_i)^2 \rangle_{SA} = \frac{(P_{NPIXEL})^2}{\Delta T_{FRAME}} \qquad (4.1)$$

$P_{NPIXEL}$ is noise power for a single array element.

For a single detector scanning FLIR

$N_T = N_{TSD}$ (number of steps for a two-dimensional scan, minimum of N).

$$\langle (\Delta A_i)^2 \rangle_{SD} = \frac{N_{TSD}(P_{NPIXEL})^2}{\Delta T_{FRAME}} \qquad (4.2)$$

For a linear scanning FLIR

$N_T = N_{TLS}$ (number of steps for a one-dimensional scan, minimum of $\sqrt{N}$ ).

$\sum_j B_{ij}^2 = 1$, since only one non-zero coefficient

$$\langle (\Delta A_i)^2 \rangle_{LS} = \frac{N_{TLS}(P_{NPIXEL})^2}{\Delta T_{FRAME}} \qquad (4.3)$$

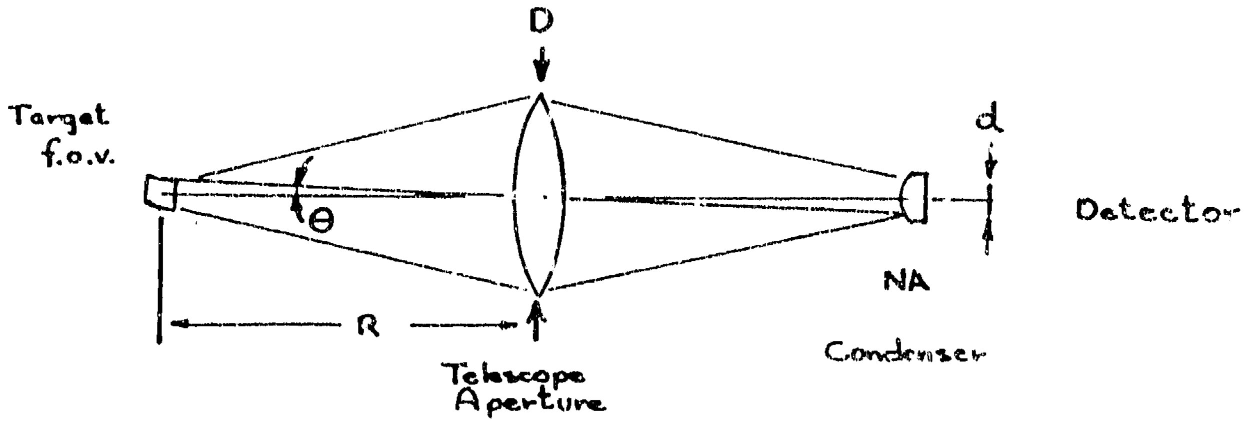
In order to make a direct comparison between the effects of the imaging techniques themselves, comparable imaging conditions must be defined. First, the frame time $\Delta T_{FRAME}$ is set to one. Then the relationship between the detector noise power for AIT must be correctly written. To cover a specific field of view, each pixel is mapped into a corresponding solid angle of target space. The mapping of angular f.o.v. in target space onto the corresponding detector area is illustrated in Figure 4.6. It is seen that for a given field angle $\Theta$, the detector size d required for proper coverage is

$$d = \frac{2D\Theta}{NA} \qquad (4.4)$$

where D is the telescope aperture diameter and NA is the numerical aperture of the detector condenser optics.

If $\Theta$ is the total imaging f.o.v., then in the FLIR systems, each detector covers the angular region corresponding to one pixel. For an N x N field, the pixel field angle is $\Theta/N$, or

$$d_{FLIR} = \frac{2D\Theta}{N(NA)} \qquad (4.5)$$



| | |
|---|---|
| R = | Range in meters |
| θ = | Telescope f.o.v. in rad |
| D = | Telescope aperture diameter |
| d = | Detector diameter |
| NA = | Condenser numerical aperture |

Detector size to match f.o.v.

$$d = \frac{2D\theta}{NA}$$

(Conservation of Étendue)

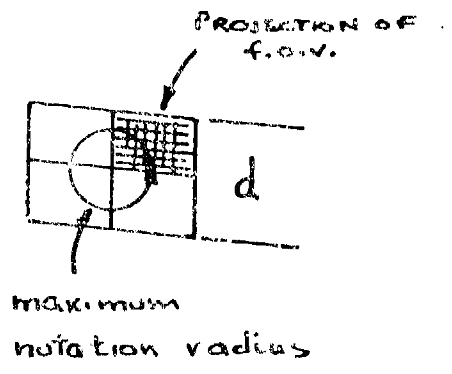


FIGURE 4.6. RELATIONSHIP BETWEEN ANGULAR F.O.V. AND DETECTOR AREA

For AIT, each detector quadrant must cover the entire f.o.v., so the AIT detector size is given by equation (4.4). Since, for IR detectors, the detector noise power is given by

$$P_{NOISE} = d^2_{IMAGER} B\ D^* \qquad (4.6)$$

where B is the signal bandwidth, and D* is the normalized detectivity of the detector.

Combining equations (4.4) - (4.6), we obtain

$$P_{NAIT} = N \cdot P_{NFLIR} \text{ (Detector or background noise limit)} \qquad (4.7)$$

In the photon noise limit, the noise is proportional to the area $A_t$ of the target, rather than the area of the detector, so

$$P_{NAIT} = A_t \cdot P_{NFLIR} \text{ (photon noise limit)} \qquad (4.8)$$

The imaging signal to noise performance of idealized FLIRs versus several versions of the AIT are collected in Table 4.2. The comparison assumes an 8 x 8 pixel field. The "15/31 Ellipse" AIT is the system studied under this program, with a mean noise factor of 1.05 obtained by averaging the noise factors of all the 64 pixels. The "ideal" AIT assumes a perfect raster scan nutation pattern to minimize noise factor, which is estimated at .25. This kind of pattern would be unsuitable for tracking.

Performance for photon noise limit can be obtained by the substitution indicated in equation (4.8).

COMPARATIVE S/N PERFORMANCE OF DIFFERENT IMAGER TYPES

NOISE POWERS VARY AS DETECTOR AREA.
AIT DETECTORS MUST COVER FULL F.O.V.,
FLIR DETECTORS COVER ONLY SINGLE PIXEL
F.O.V.

$\therefore$ FOR $N = 64$ PIXELS

$P_{NAIT} = \sqrt{N}\, P_{NPIXEL} = 8\, P_{NPIXEL} = 8\, P_N$

$\Delta T_{FRAME} = 1$

| IMAGER | # OF DETECTORS | $\overline{\sum_j B_{ij}^2}$ | $N_T$ | $P_{NOISE}$ | $\sqrt{\sum_j B_{ij}^2 N_T P_{NOISE}^2} = \sqrt{N_p} \sim \sigma$ |
|---|---|---|---|---|---|
| STARING ARRAY | 64 | 1 | 1 | $P_N$ | $P_N$ |
| LINEAR SCAN FLIR | 8 | 1 | 8 | $P_N$ | $2.82\ P_N$ |
| SINGLE ELEMENT FLIR | 1 | 1 | 64 | $P_N$ | $8P_N$ |
| AIT (15/31 ELLIPSE) | 4 | 1.05 | 1860 | $8P_N$ | $353.5\ P_N$ |
| "IDEAL" AIT | 4 | (.25) | 256 | $8P_N$ | $64\ P_N$ |

TABLE 4.2

## 4.6 Simulation Results

The reconstruction matrix can be tested for image formation using the combination of programs SIMDSK and IMAGE. The input to the SIMDSK program, which generates the simulated waveforms is defined as a numerical intensity file. An example is shown in Figure 4.7, which is the input file for a 2 x 4 pixel horizontal bar, located at coordinates 0, 0, and having unit intensity. The output of SIMDSK is a file containing the nutation waveforms corresponding to the input image. Figure 4.8 shows a plot of the waveforms, unpacked to show the outputs of quadrants 1 - 4 individually.

These output waveforms can then be used as inputs to the IMAGE program which multiplies the detector outputs by the reconstruction matrix and displays the results. The pseudo-color output for the simulated horizontal bar is shown in Figure 4.9.

The results shown are essentially perfect. This is due partly of course to the high accuracy of the reconstruction algorithm, but also to the fact that the input object was defined in perfect registration with the pixel field, so that no effects of spatial quantization of the image are encountered.

A more realistic case is the reconstruction of the same bar rotated $45^0$, as shown in Figure 4.10. Here the effects of spatial quantization are obvious, and the results are more representative of image coding at this coarse resolution.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

FIGURE 4.7

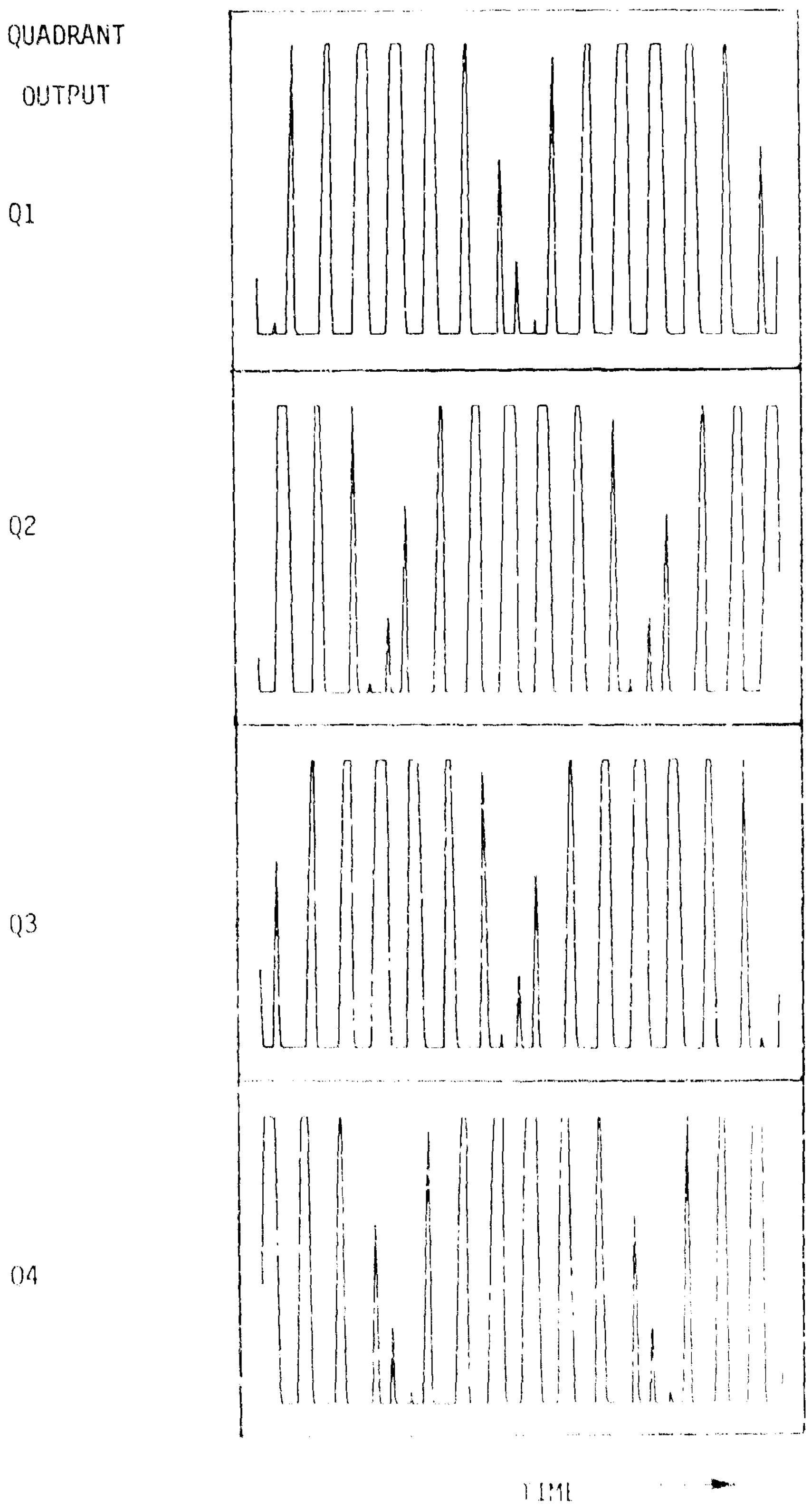
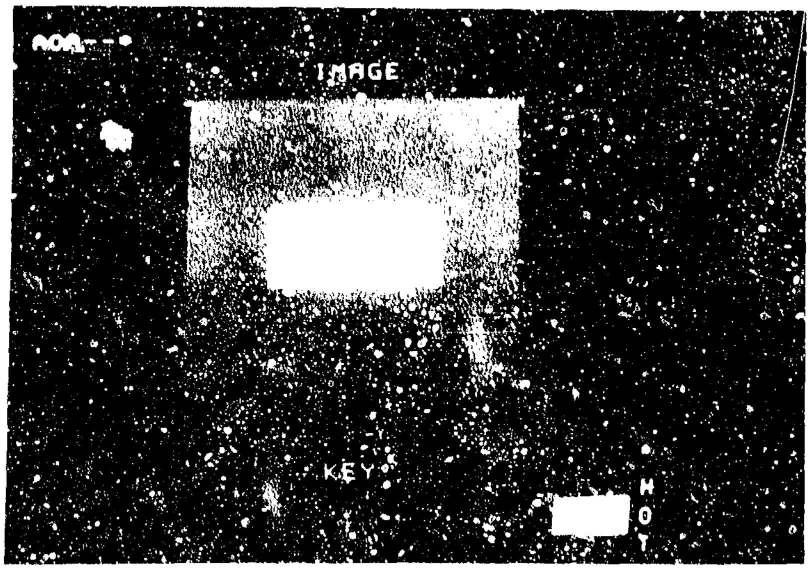
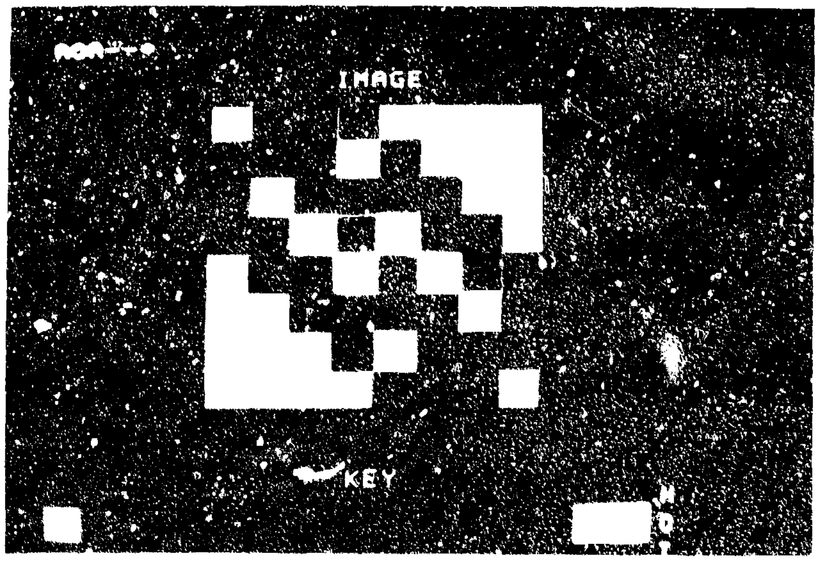


FIGURE 4.8



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -4 | 2 | -1 | 0 | 1 | -1 | 2 | -4 |
| 2 | -1 | 0 | 0 | 0 | 0 | -1 | 2 |
| -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 |
| 1 | -1 | 205 | 205 | 206 | 204 | 0 | 0 |
| 0 | 0 | 204 | 206 | 205 | 205 | -1 | 1 |
| 0 | -1 | 0 | -1 | 0 | -1 | 0 | -1 |
| 2 | -1 | 0 | 0 | 0 | 0 | -1 | 2 |
| -4 | 2 | -1 | 1 | 0 | -1 | 2 | -4 |

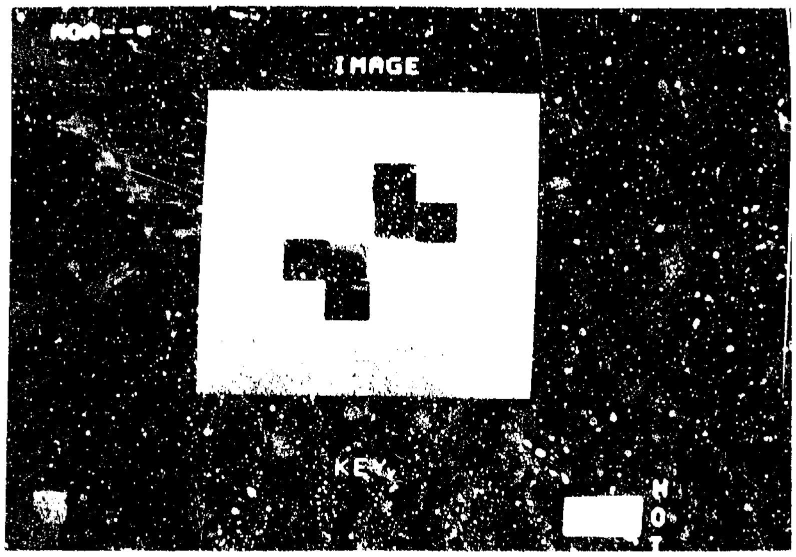
FIGURE 4.9



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -3 | 10 | 54 | 10 | -6 | 2 | 2 | 3 |
| 11 | 51 | 56 | 61 | 15 | -6 | -6 | 4 |
| 50 | 62 | 45 | 55 | 53 | 15 | 7 | [illegible] |
| 13 | 52 | 69 | 34 | 70 | [illegible] | 19 | [illegible] |
| -6 | 19 | 43 | 70 | 34 | 69 | 52 | 13 |
| 0 | 2 | 15 | 53 | 55 | 45 | 62 | [illegible] |
| 4 | -5 | -6 | 15 | 61 | 56 | 51 | 11 |
| -3 | 2 | 2 | -6 | 10 | 54 | 10 | -3 |

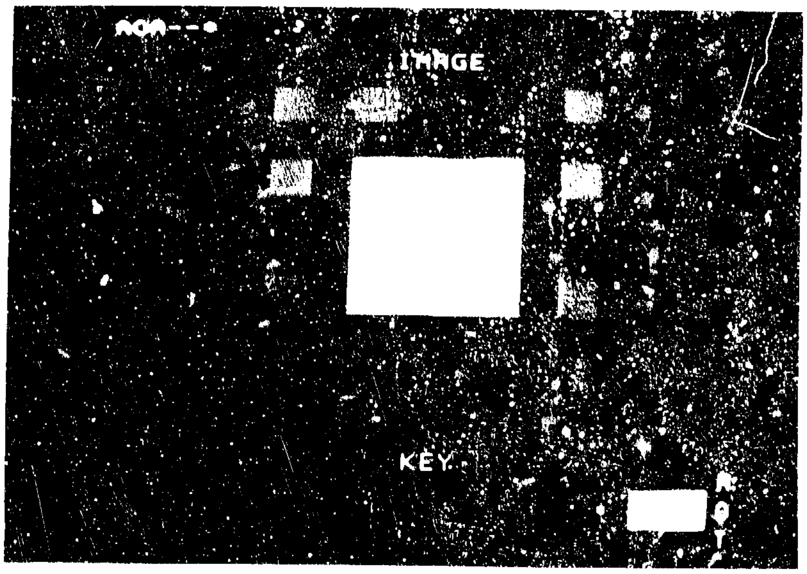
FIGURE 4.10

The effects of quantization become more apparent in reconstruction of continuous-intensity objects. Figure 4.11 shows a small Gaussian profile spot reconstruction, and Figure 4.12 shows a square with Gaussian smoothed edges. As well as the coarseness of reconstruction, an axis of diagonal symmetry is apparent in what are defined to be circular and four-fold symmetric objects. This feature is an artifact of the sampling pattern, which, especially near the center of the field, has non-uniform sampling reflecting the same diagonal symmetry.



|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| 4 | 2 | 1 | 1 | 2 | 2 | 2 | -4 |
| 3 | 1 | 4 | 18 | 5 | 3 | -4 | 3 |
| -1 | 0 | 34 | 56 | 90 | 17 | 10 | -2 |
| 0 | 10 | 66 | 227 | 195 | 88 | 7 | 2 |
| 2 | 2 | 87 | 195 | 227 | 66 | 10 | 0 |
| -2 | 3 | 17 | 90 | 56 | 34 | 0 | -1 |
| 3 | -4 | 3 | 5 | 18 | -4 | 1 | 2 |
| -4 | 2 | 2 | -2 | 1 | -1 | 2 | -4 |

FIGURE 4.11



|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| -1 | 6 | 3 | 4 | 5 | 4 | 5 | [illegible] |
| [illegible] | 26 | 39 | [illegible] | 33 | 33 | 26 | [illegible] |
| 0 | 41 | 53 | 52 | 50 | 54 | 41 | [illegible] |
| 6 | 37 | 51 | 49 | 50 | 51 | [illegible] | [illegible] |
| 5 | 38 | 53 | 50 | 48 | [illegible] | 37 | 4 |
| 2 | 42 | 54 | 50 | 52 | 53 | 41 | [illegible] |
| 8 | 26 | 39 | 38 | 33 | 39 | 26 | [illegible] |
| -1 | 5 | 4 | 5 | 4 | 3 | 8 | [illegible] |

FIGURE 4.12

## 4.7 Algorithm Parameter Sensitivity

The simulation codes were used to test the sensitivity of the reconstruction algorithm to small variations in system parameters of engineering importance. Tested were sensitivities to

1) Waveform shift. Routines were written to numerically shift the detector waveforms in time. This was done to simulate the effects of phase shifts in real detector circuits and phase error in synchronism between the nutation scan and sampling circuitry.

2) Waveform smoothing. Simulates the effects of bandlimiting in the readout electronics.

3) DC level-shifting. Tests the effects of AC coupling the detector outputs on image reconstruction.

The routines used to perform these operations transformed the detector vector disk file and stored the conditioned waveform in a local memory buffer. The codes were organized as signal conditioning options on a menu at the start of the IMAGE program.

### 4.7.1 Waveform shift

The detector vectors are packed into a linearly organized file. Properly unpacked, the detector values can be treated as a two-dimensional array, so that any element d can be labeled as $d_{(q,t)}$, where q = 1 - 4 (quadrant number) and t = 1 - 465 (time sample number).

A phase shift of one full sample can be accomplished by the cyclic permutation:

$$d(q,t) = d(q,t+1) \qquad t = 1 - 464 \qquad \text{positive phase shift}$$
$$d(q,465) = d(q,1)$$

$$d(q,t) = d(q,t-1) \qquad t = 2,465 \qquad \text{negative phase shift}$$
$$d(q,1) = d(q,465)$$

A phase shift of less than one sample is done by sample interpolation:

$$d(q,t) = d(q,t) + s(d(q,t+1) - d(q,t)) \quad t = 1,464$$
$$d(q,465) = d(q,465) + s(d(q,1) - d(q,465))$$

positive phase shift

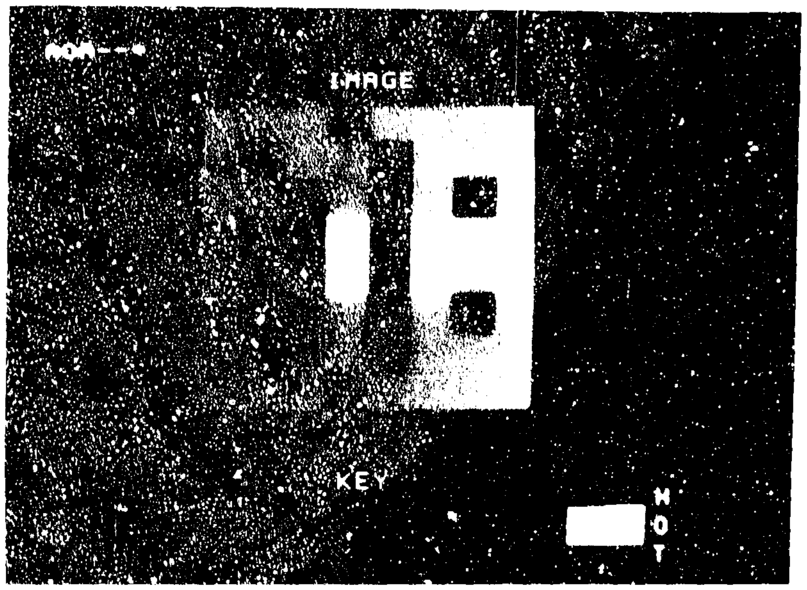
$$d(q,t) = d(q,t) + s(d(q,t-1) - d(q,t)) \quad t = 2,465$$
$$d(q,1) = d(q,1) + s(d(q,465) - d(q,1))$$

negative phase shift

By iterative reconstruction of a simulated input detector vector conditioned by various phase shifts, the sensitivity to waveform phase was determined. A shift of a full time sample was found to completely destroy the reconstruction. The largest phase shift that can be used without unacceptable image degradation is approximately .1 time sample. Figure 4.13 shows a simulated horizontal bar phase shifted by .1 time sample. The total power in the image is reduced by about 1%.

### 4.7.2 Waveform smoothing

In order to test the effect of waveform smoothing as might be caused by electronic bandwidth limitations, a simple digital smoothing filter was used:



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -2 | -10 | 2 | [illegible] | [illegible] | -4 | 14 | [illegible] |
| -5 | 30 | -14 | -46 | 44 | 14 | -34 | [illegible] |
| [illegible] | 53 | 20 | -34 | 44 | [illegible] | [illegible] | [illegible] |
| [illegible] | 41 | 131 | [illegible] | [illegible] | 24 | [illegible] | [illegible] |
| [illegible] | 34 | [illegible] | 262 | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | -49 | 64 | -24 | [illegible] | [illegible] | [illegible] | [illegible] |
| -5 | 30 | -14 | -47 | 48 | 12 | -34 | [illegible] |
| -2 | -10 | 2 | 32 | [illegible] | -4 | 14 | [illegible] |

FIGURE 4.13

$$d(q,t) = \tfrac{1}{4}(d(q,t-1) + 2d(q,t) + d(q,t+1)) \quad t + 2,464$$

$$d(q,1) = \tfrac{1}{4}(d(q,465) + 2d(q,1) + d(q,2))$$

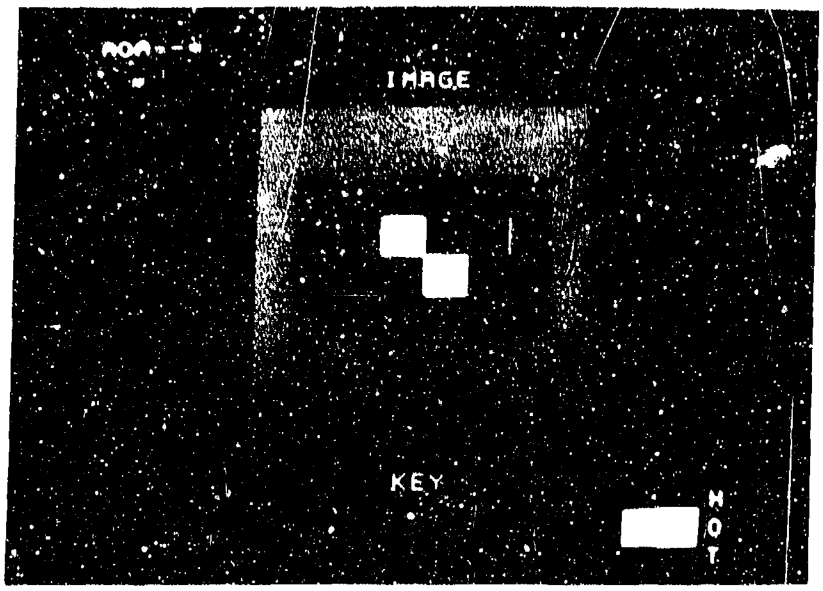
$$d(q,465) = \tfrac{1}{4}(d(q,464) + 2d(q,465) + d(q,1))$$

The result of smoothing the waveform is a smoothing of the reconstructed image. Figure 4.14 shows the results of passing the detector vector through the smoothing filter once, while Figure 4.15 is the same image after three passes of the waveform smoothing filter. The major effect is a continued loss of image resolution.

### 4.7.3 Level shifting

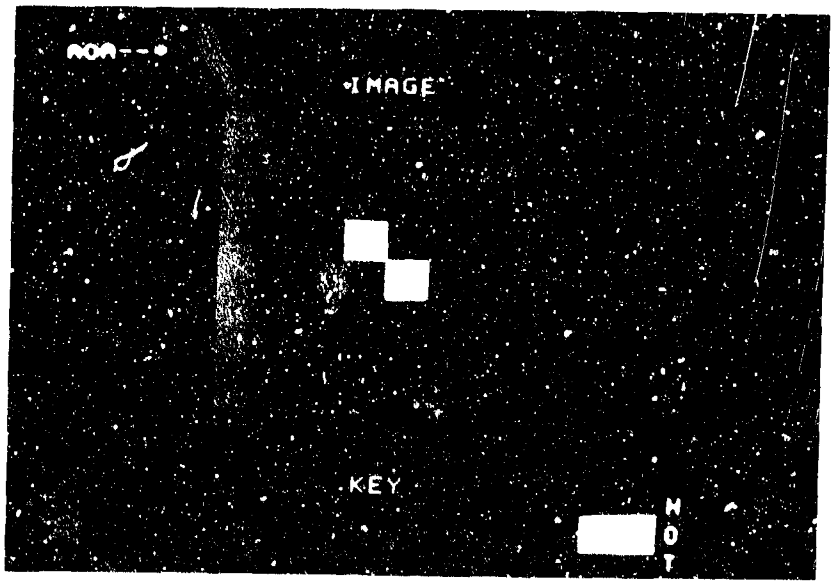
The AIT reconstruction algorithm was defined with the implicit assumption that the input detector samples were specified positive definite with respect to an absolute zero level. From an engineering point of view, this implies a DC coupled detection system. In an AC coupled system the detector waveform has positive and negative value extremes such that the RMS value of the waveform is zero. This results in an offset or negative DC pedestal on the waveform. Figure 4.16 shows the results of level shifting the simulated waveforms for the horizontal bar negative by one-half the peak waveform value. The reconstruction of the bar is unaffected but artifacts are created in the corners of the reconstruction field.

The DC offset of the waveform is physically equivalent to superposing the target bar on a uniform (negative) intensity, including intensity outside the field of view of the algorithm as defined by the pixel locations which generated the forward matrix.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -4 | 3 | -1 | 1 | 1 | 1 | 2 | -4 |
| 4 | -4 | -5 | 5 | -11 | -2 | -6 | 5 |
| -5 | 16 | 25 | 19 | 53 | 19 | 17 | -4 |
| -2 | 30 | 134 | 201 | 142 | 166 | 20 | -2 |
| -2 | 20 | 166 | 142 | 201 | 134 | 30 | -2 |
| -4 | 17 | 19 | 53 | 19 | 25 | 16 | -5 |
| 5 | -6 | -2 | -11 | 5 | -5 | -4 | 4 |
| -4 | 2 | 1 | 1 | 1 | -1 | 3 | -4 |

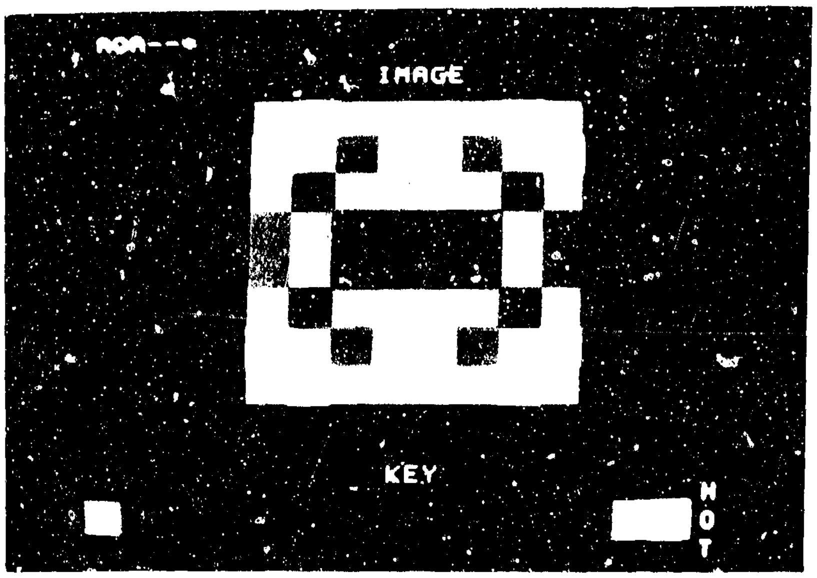
FIGURE 4.14

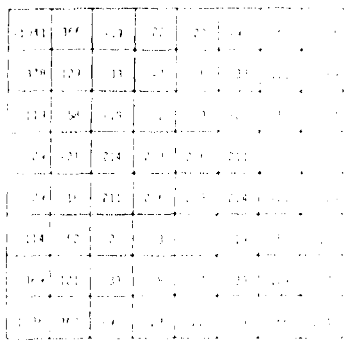


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -4 | 0 | -1 | -2 | 1 | 2 | 0 | -4 |
| 3 | 12 | -8 | 21 | -13 | -1 | 9 | 5 |
| 6 | 25 | 45 | 21 | 78 | 44 | 22 | 8 |
| -4 | 41 | 86 | 184 | 85 | 126 | 31 | -4 |
| -4 | 31 | 126 | 85 | 184 | 86 | 41 | -4 |
| 8 | 22 | 43 | 77 | 21 | 45 | 25 | 6 |
| 5 | 9 | -1 | -13 | 21 | -8 | 12 | 3 |
| -4 | 0 | 2 | 1 | -2 | -1 | 0 | -4 |

FIGURE 4.15

( )





| | | | | | | | |
|---|---|---|---|---|---|---|---|
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | | |
| [illegible] | [illegible] | 13 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | | |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

FIGURE 4.16

This power outside the reconstruction field is aliased into the field, forming the corner artifacts.

In order to employ AC coupling in the AIT imaging system, a simple DC restoration step was used to re-reference the waveform to zero absolute minimum. This allows effective operation of the reconstruction in either AC coupled or DC coupled systems exhibiting offset drift.

## 5.0 SOFTWARE SIMULATION: TRACKING

The generalization of the original SCS tracking algorithm to accommodate the more complex elliptical nutation pattern necessitated the development of new software to perform and test the algorithm and to test the tracking concept.

The procedure for implementing the AIT tracker algorithm defined in Section 3 is shown in the flow diagram of Figure 5.1. The generation of all the parameters required for execution of the algorithm was coded, so that, given a description of the target object and the nutation minor cycles, the mask functions, gains, and offset corrections were calculated.

Initial programs written to test the tracking algorithm created detector vectors by assuming the target to be a point source with a given displacement relative to the center of the field of view. These programs would also determine the "allowable arcs" to be used for a given object size (even though the target was considered a point source). Using these detector vectors and allowable arc definitions, the programs would also calculate the gain and offset for each tracker cycle. Algorithm consistency was initially verified by using the gains and offsets calculated for a given object displacement in determining the positions of point sources with different displacements. Thereby, the sensitivity of the algorithm to sets of displacements measured across sets of gains and offsets was examined.

The 15/31 collapsing ellipse nutation pattern was divided into minor cycles as shown in Figure 5.2 - 5.4.

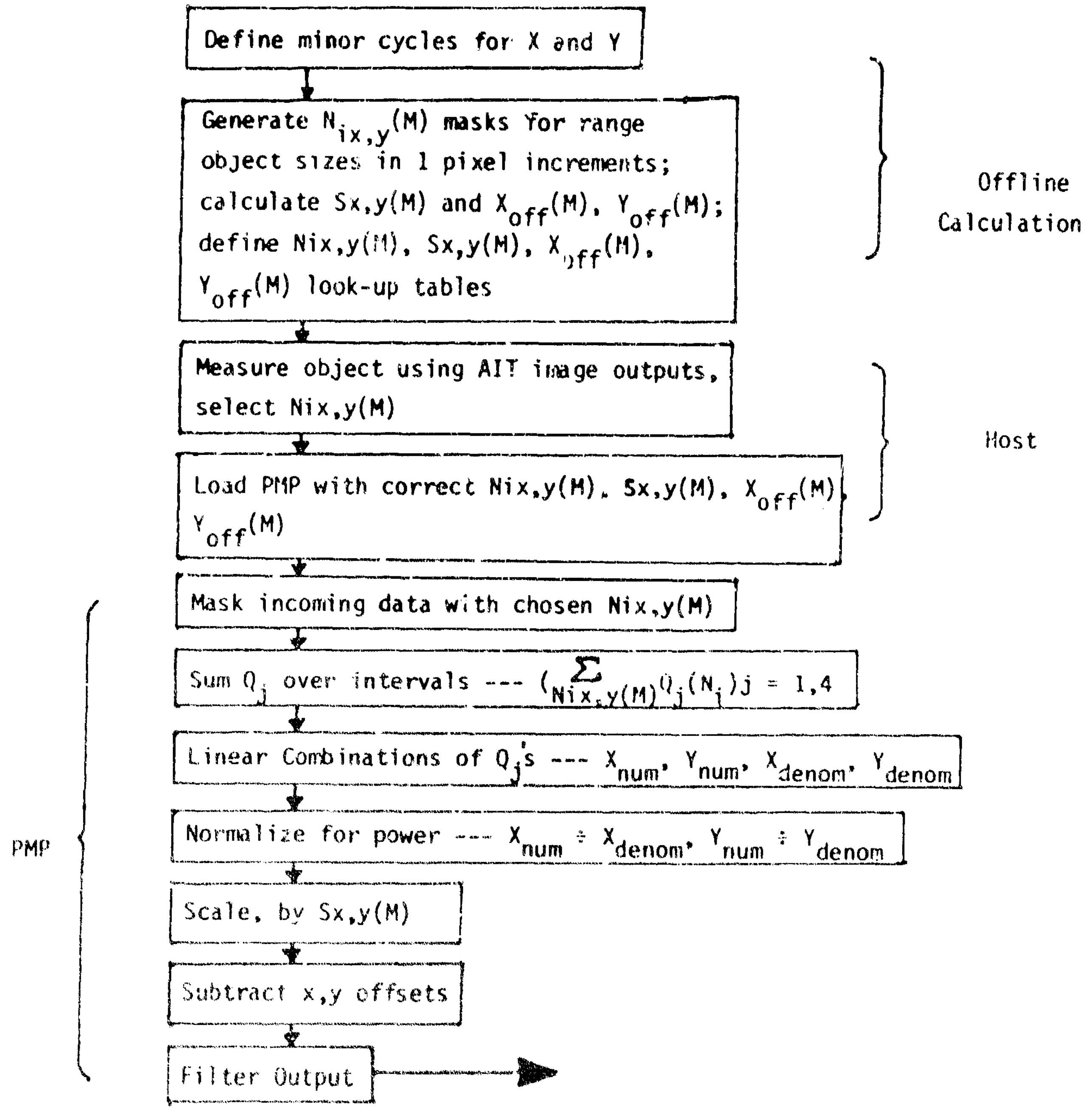

Define minor cycles for X and Y
Generate $N_{ix,y}(M)$ masks for range object sizes in 1 pixel increments; calculate Sx,y(M) and $X_{off}(M)$, $Y_{off}(M)$; define Nix,y(M), Sx,y(M), $X_{off}(M)$, $Y_{off}(M)$ look-up tables
Offline Calculation
Measure object using AIT image outputs, select Nix,y(M)
Load PMP with correct Nix,y(M), Sx,y(M), $X_{off}(M)$, $Y_{off}(M)$
Host
Mask incoming data with chosen Nix,y(M)
Sum $Q_j$ over intervals --- $(\sum_{Nix,y(M)} Q_j(N_i)) j = 1,4$
Linear Combinations of $Q_j$'s --- $X_{num}$, $Y_{num}$, $X_{denom}$, $Y_{denom}$
Normalize for power --- $X_{num} \div X_{denom}$, $Y_{num} \div Y_{denom}$
Scale, by Sx,y(M)
Subtract x,y offsets
Filter Output
PMP

FIGURE 5.1

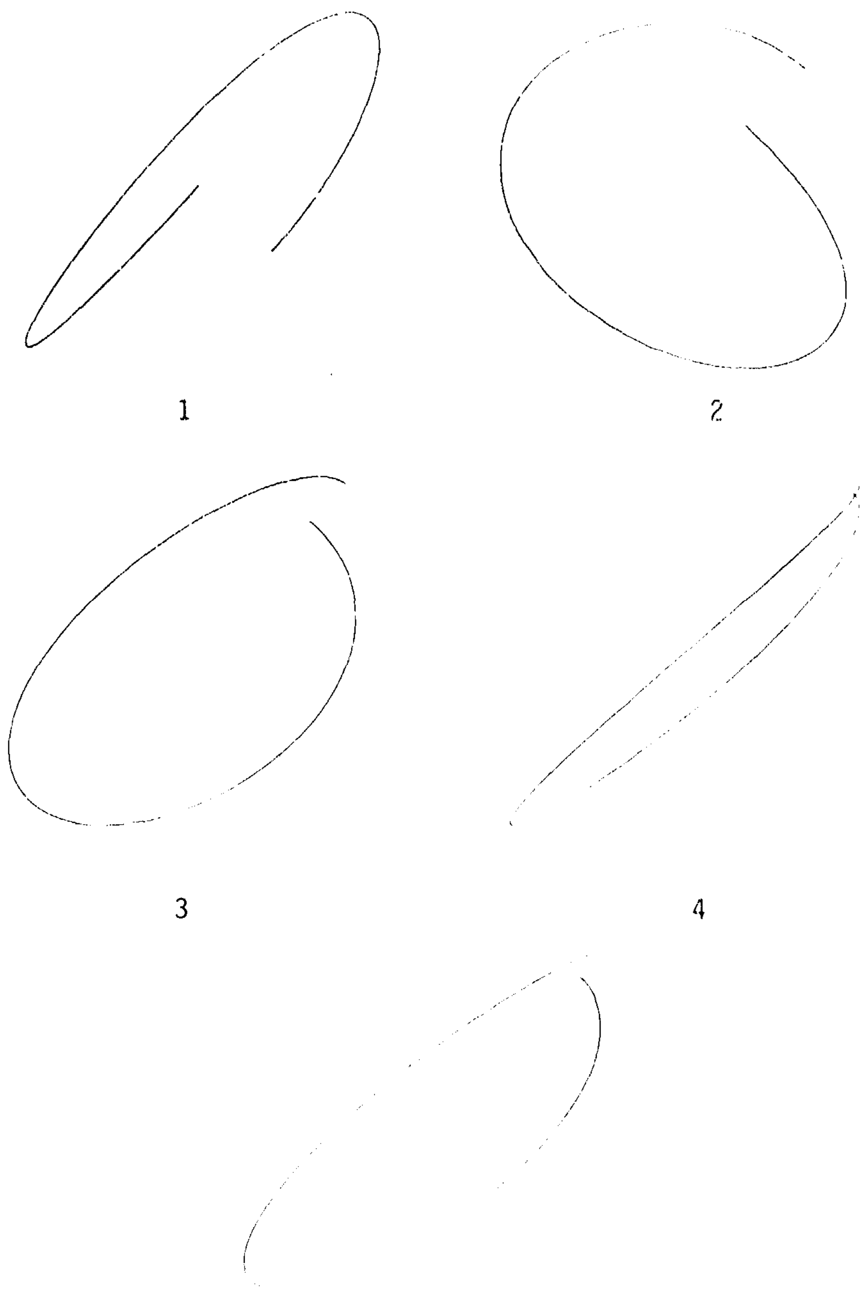


FIGURE 5.2 AIT MINOR CYCLES

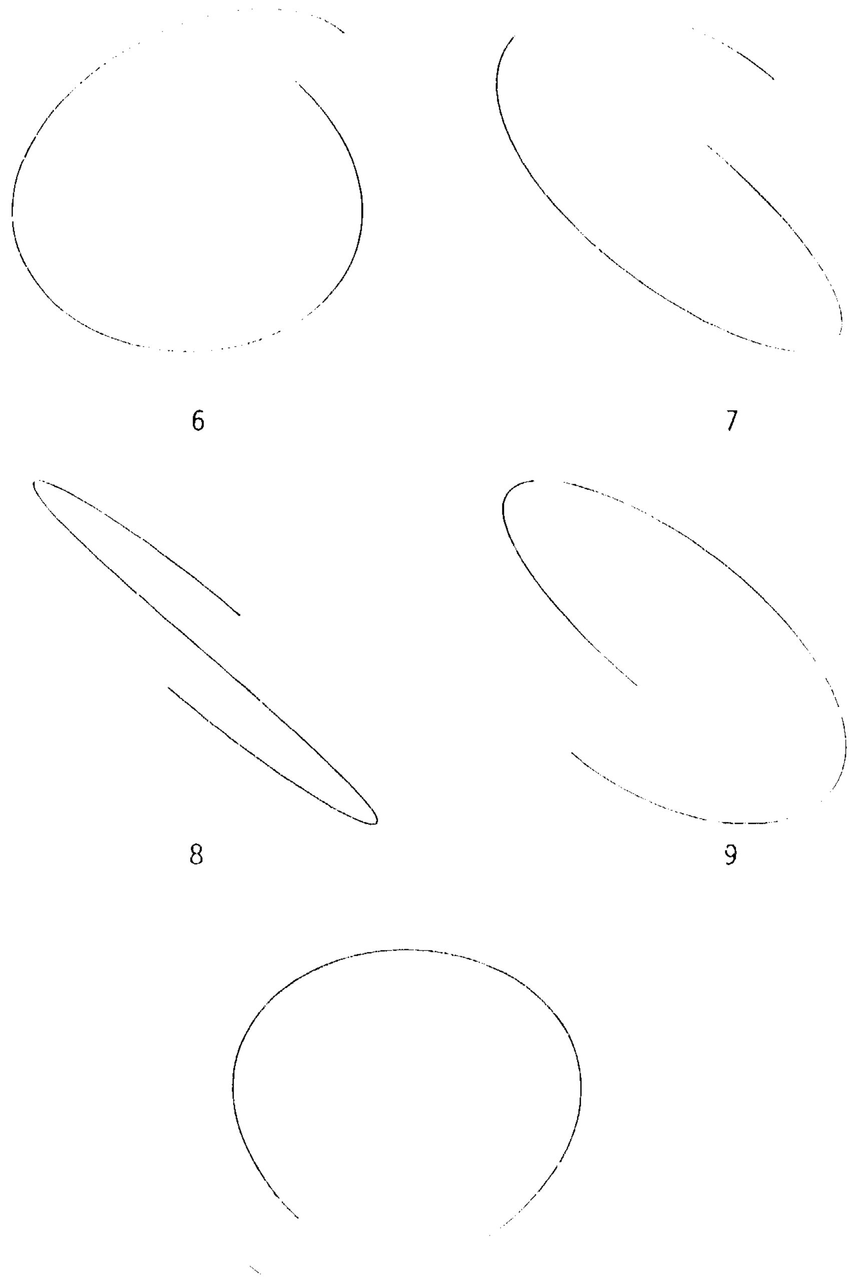


FIGURE 5.3 AIT MINOR CYCLES (cont'd)

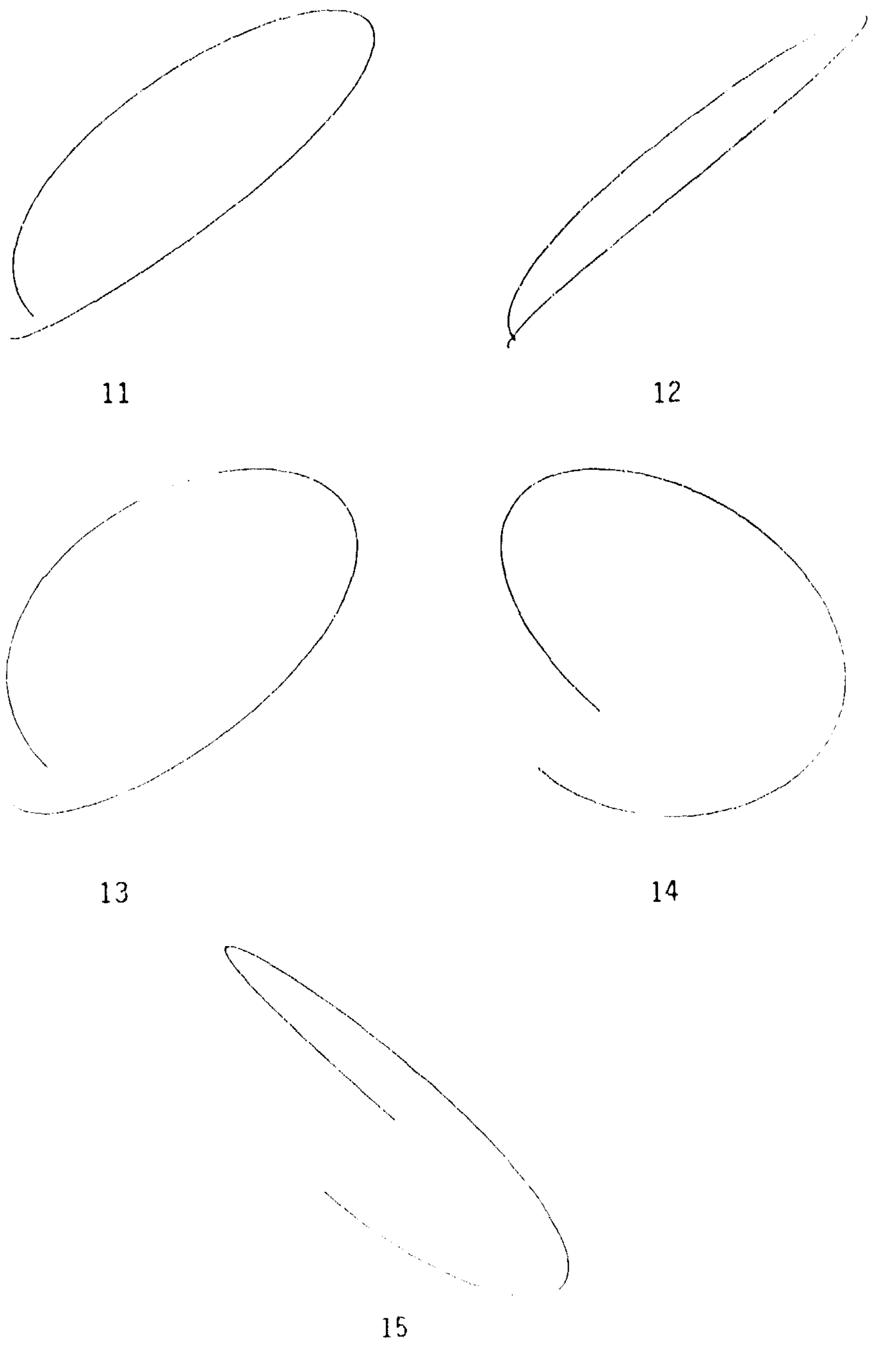


FIGURE 5.4 AIT MINOR CYCLES (cont'd)

## 5.1 Tracking Performance

Transfer functions for an AIT tracker algorithm defined using a 2 pixel sized generator object were simulated. Small displacement response was investigated using various test object sizes, while large displacement response was investigated over the entire 8 x 8 field using single pixel test objects.

Figure 5.5 shows the tracker transfer function for small displacements. In this simulation, a roughly two-pixel diameter Gaussian intensity profile spot was translated along the x-axis in the range -.75 pixels to +.75 pixels, and the tracker response was calculated for several different minor cycles. The results are excellent. The residual offset spread for the transfer curves through zero is less than .06 pixel size, or less than .03 of the 2 pixel spot diameter. Furthermore, the average of the minor cycle offsets is found to be zero to within the accuracy of the simulation. The algorithm displays no residual bias for averaging times of one full pattern length or longer.

In Figure 5.6 the same calculation is repeated using a 4 x 4 pixel square test object with Gaussian smoothed edges. This object is larger by twice than the limit allowed by the definition of the algorithm. Nevertheless, operation is very good, with about .1 pixel offset spread and + 10% effective gain variation in the transfer function for different minor cycles. Again the average offset for the minor cycles is zero. This result shows that tracker performance degrades gracefully as the object size exceeds that for

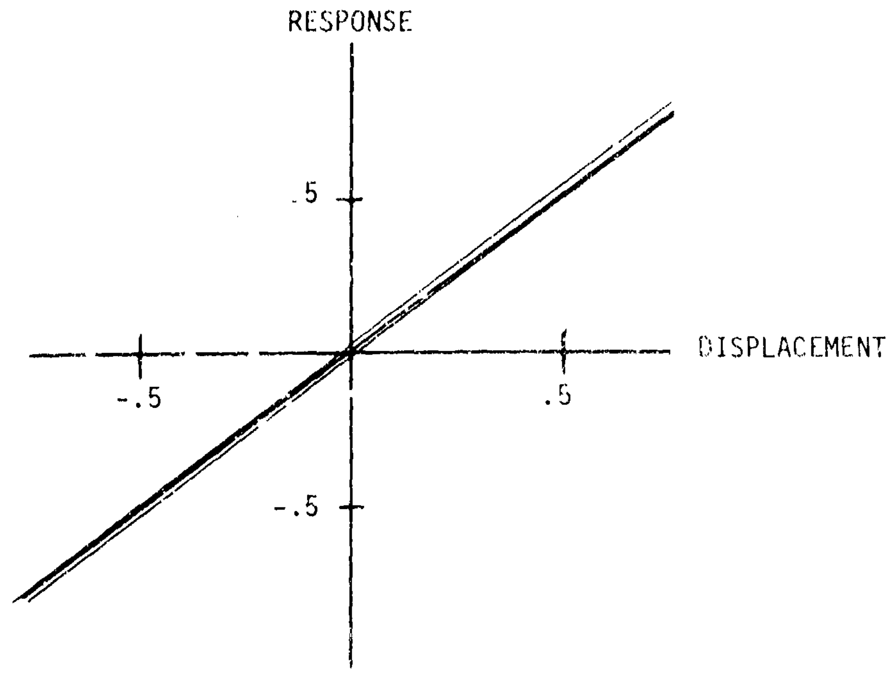


FIGURE 5.5

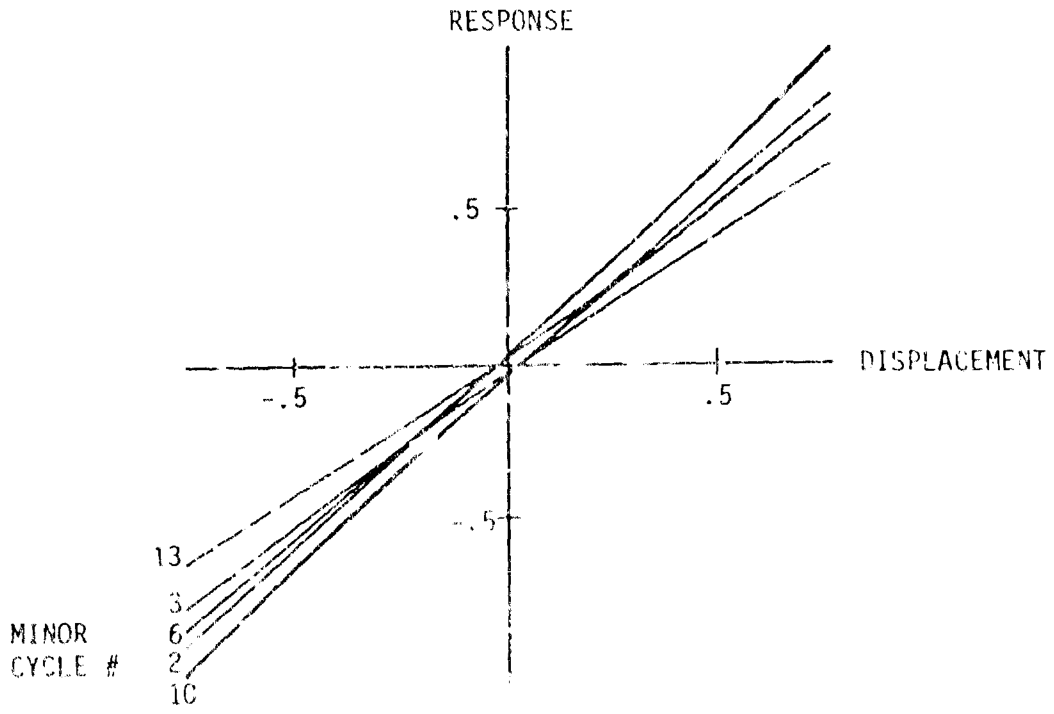


FIGURE 5.6

which the algorithm coefficients are optimized, which is a very important practical feature.

The large signal response curves also show useful performance features. Figure 5.7 shows the tracker response to a one-pixel size spot translated between the coordinates x = $\pm$ 3.5 pixels. The path is displaced to offset y = .5 pixel. Even at the edges of the reconstruction field the slope change are not large.

In Figure 5.8, the object is again translated in the x-direction, but with a displacement of 2.5 pixels along the y-axis. Even at the extremes of the displacement the curves are monotonic. This property holds throughout the entire field; i.e. there are no slope reversals that would result in any system instability. The algorithm under all conditions responds with the correct sign of error signal, and in any closed-loop application would drive the system towards null, where the ultimate response is excellent.

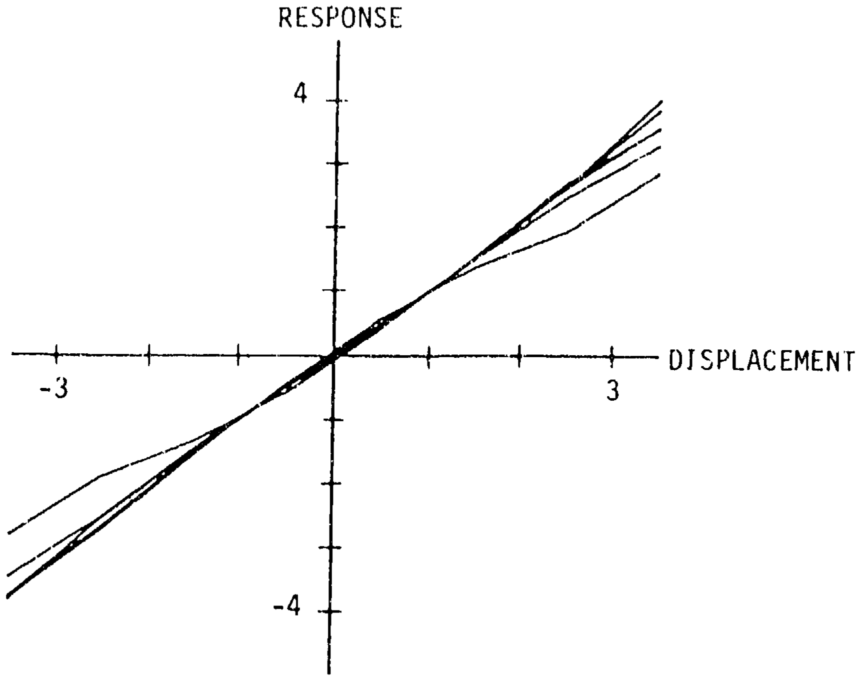


FIGURE 5.7

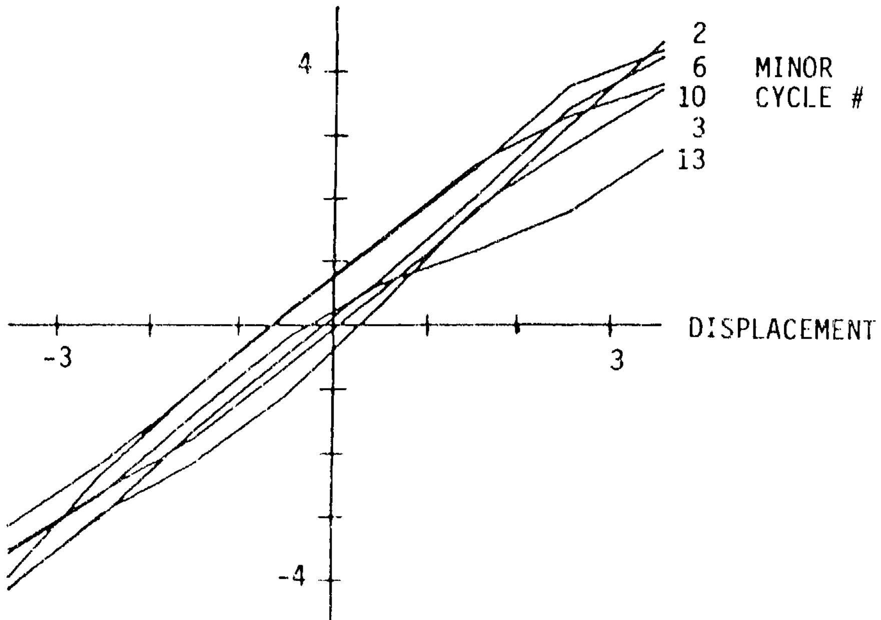


FIGURE 5.8

## 6.0 BREADBOARD HARDWARE: OPTICAL SYSTEM

As part of the AIT program, a breadboard optical system was designed and constructed to generate nutated detector waveforms from real target images. The images were produced by a target simulator in which target size, shape, intensity, position and rotation could be controlled. Both systems were designed to operate at either visible or infrared wavelengths.

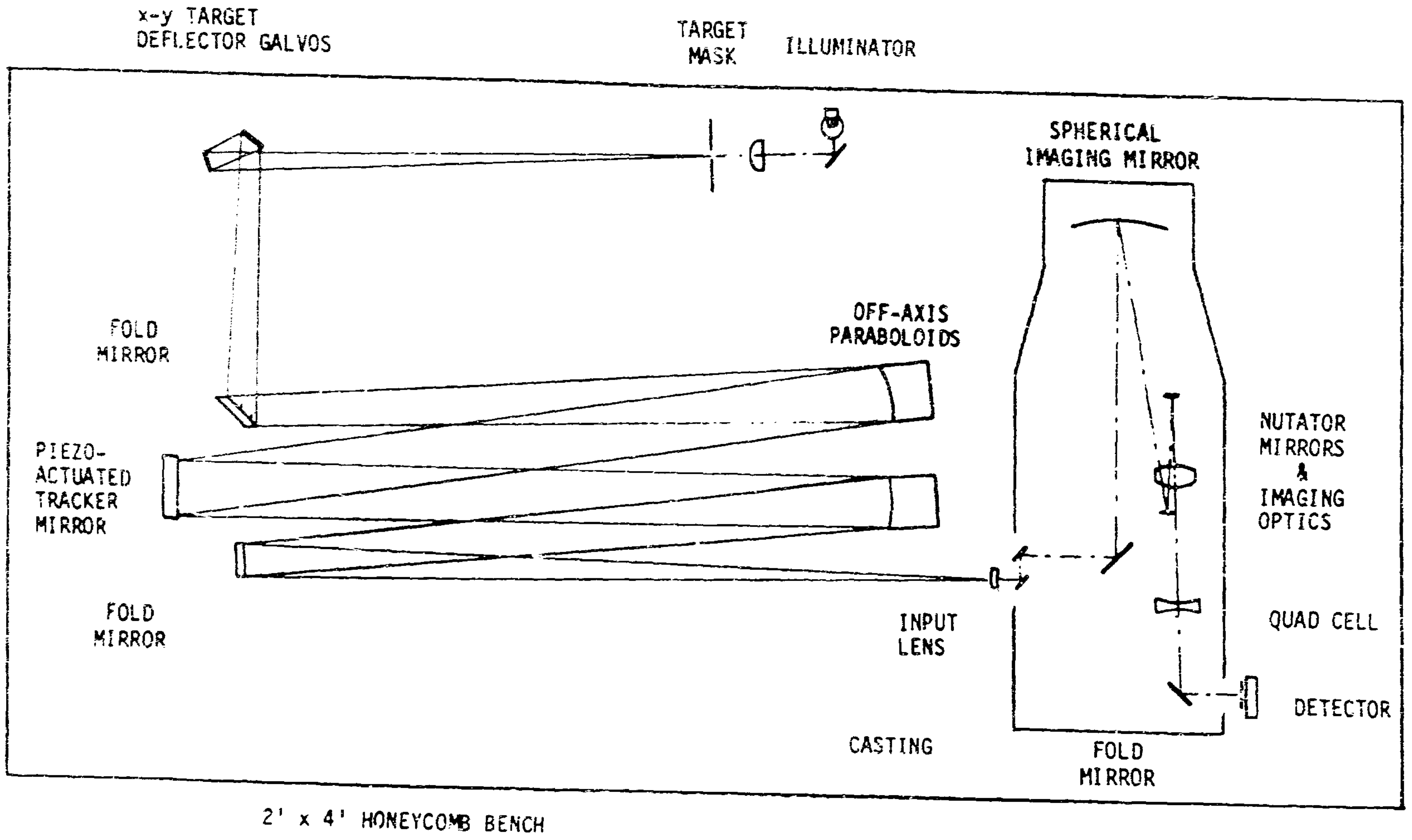
### 6.1 Optical Layout

Figure 6.1 is a diagram of the AIT optical head and target simulator system, drawn approximately to scale. A photograph of the set-up is shown in Figure 6.2.

The simulated target originates as a target mask, shown at top center in Figure 6.1. For visible operation, the target could be a continuous-tone 35 mm transparency or a thin chemically-etched copper mask. For IR operation, only the copper masks would be used. The target is back-illuminated by a simple tungsten lamp and condenser arrangement; in the IR, the tungsten lamp would be replaced by a black-body source. The target mask itself is mounted on a stepper-motor driven rotary stage, allowing target rotation to be simulated.

The expanding beam from the target projector passes through a pair of galvanometer mirror deflectors, shown in the upper left corner of the figure. The deflectors can impose electrically controlled tip-tilt on the beam, inducing apparent motion of the target object.

AIT TARGET SIMULATOR — OPTICAL BREADBOARD



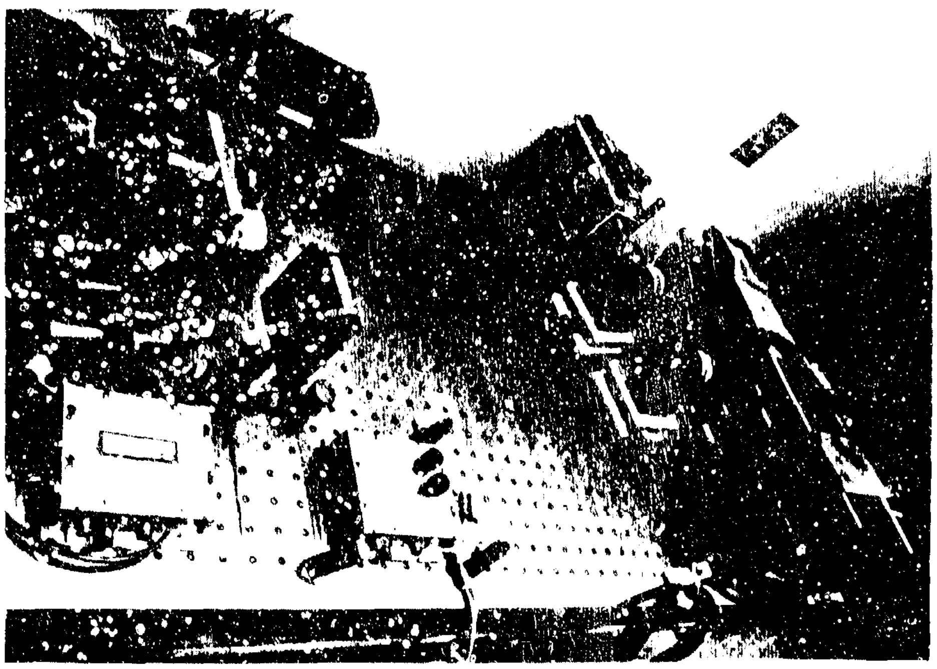


FIGURE 6.2

The beam is folded and sent to an off-axis paraboloid, which collimates the beam. The beam is then reflected from a flat mirror mounted on piezoelectric actuators. This mirror can also cause apparent target motion, or it can be used as a steering mirror in closed-loop tracking experiments.

From the piezoelectric tracker mirror the beam is reflected to another off-axis paraboloid, which brings the beam to a focus at the input of the optical head after reflection from a final fold mirror.

The optical train of the optical head must provide for the following:

1) The input must be imaged on the output quad cell detector.

2) The nutation pattern must be impressed on the input beam.

3) Means for adding a reference to the input beam must be available.

4) Means for steering the input beam from the inside of the instrument in an accurate grid pattern, with an eye to generation of an instrumental back matrix (see Section 2.4.2) is desirable.

The reference system was designed for IR use of the breadboard, and was not installed in visible operation. The first small fold mirror following the optical head input lens in Figure 6.1 would be replaces by a ZnSe beam combiner. A small black body and pinhole system, switched on and off by a flip mirror, was designed and constructed to inject a reference beam at this point.

The large spherical mirror, shown top right in the figure, is the primary element for re-imaging the input beam on the detector. It is equipped with motor-driven micrometer actuators and a precision tilt sensor, so that it can be accurately positioned electronically for generation of an instrumental back matrix.

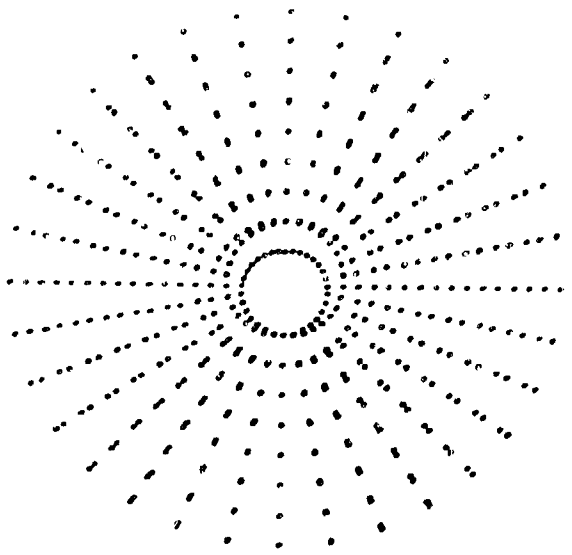
In this system, nutation is performed by a set of galvanometer scan mirrors, which separately impose the x and y nutation deflections. In order that the pupil be imaged on each mirror without intervening optical elements, an anamorphic field lens is placed at the input of instrument. This causes the pupil image to be separated to form to orthogonal line focii, coinciding with the rotation axis of each nutator mirror. A set of output lenses forms the nutated target image at an appropriate magnification on the silicon quad cell detector.

## 6.2 Nutation

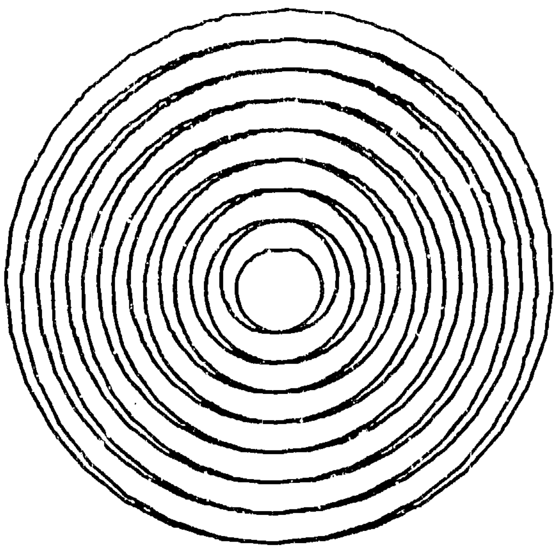
The AIT tracker-imager requires an optical scan pattern of sufficient sampling density to meet the spatial Nyquist criteria. The rule-of-thumb for a quad cell-based AIT is that the quad cell axes must sample each pixel at four distinct positions for alias-free reconstruction to be achieved. The scan pattern used in Phase I was a circular spiral which contracts linearly in time from its maximum to minimum diameter, where the latter is one-eighth the former. This is called the linear spiral. The spiral takes eight revolutions to go from maximum to minimum diameter and eight more to return to maximum, for a total of sixteen (16) revolutions per image cycle. During each revolution, thirty-two (32) time samples are taken in each quadrant, for a total of 32 x 16 x 4 = 2,048 data points per image.

The resulting sampling apttern is shown in Figure 6.3a. The spoke-like appearance of the pattern results from the use of uniformly spaced time samples. For the same reason, the sampling density is much higher near the center of the pattern. This inefficient use of the nutation time causes the exterior pixels to be marginally sampled while the interior is over-sampled. Figure 6.3b shows the path described by the nutation device over the pixel field.

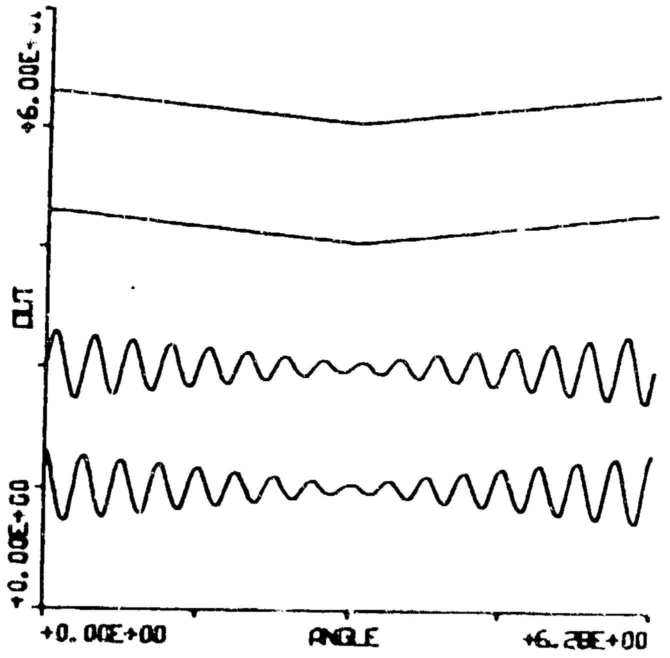
The x- and y-drive waveforms for this pattern are shown as the bottom two traces of Figure 6.3c. The signals are sine and cosine waves multiplied by linear shaped envelope functions, which are shown for x and y as the first and second traces respectively.



a.



b.



c.

FIGURE 6.3

For the experimental work done in Phase I, a two-axis piezoelectric deflector driven by a digitally programmable waveform generator was used to produce this nutation pattern. It was adequate for demonstration purposes but is too slow (approximately 250 Hz useful frequency range) to be usable in a field instrument.

Previous work on the related $I^3$ sensor utilized resonant galvanometer scan mirrors to provide a circular nutation pattern with constant amplitude at 10 kHz scan rates. Using two mirrors (one per axis) in phase quadrature, the nutation can be controlled to better than 1% in both amplitude and phase.

However, the resonant scan mirrors are designed to be high-Q devices (Q is greater than 1000) in order to achieve the required deflections at high speeds. At 10 kHz, this implies a control bandwidth of less than 100 Hz. A Fourier analysis of the triangle wave envelope function of Fiugre 6.3c yields frequency components at the fundamental, $(10^4/16) = 625$ Hz, with higher harmonic terms at frequencies $(625 \pm 625 \times n)$ Hz, n odd, whose amplitude decreases as $1/n^2$. This means that the mirrors must respond to frequency components several kilohertz above and below the 10 kHz center frequency in order to approximate the required deflection waveforms (Figure 6.3c, bottom). This is precluded by the high mirror Q. The conclusion is that a pair of resonant galvos cannot generate the necessary AII nutation pattern.

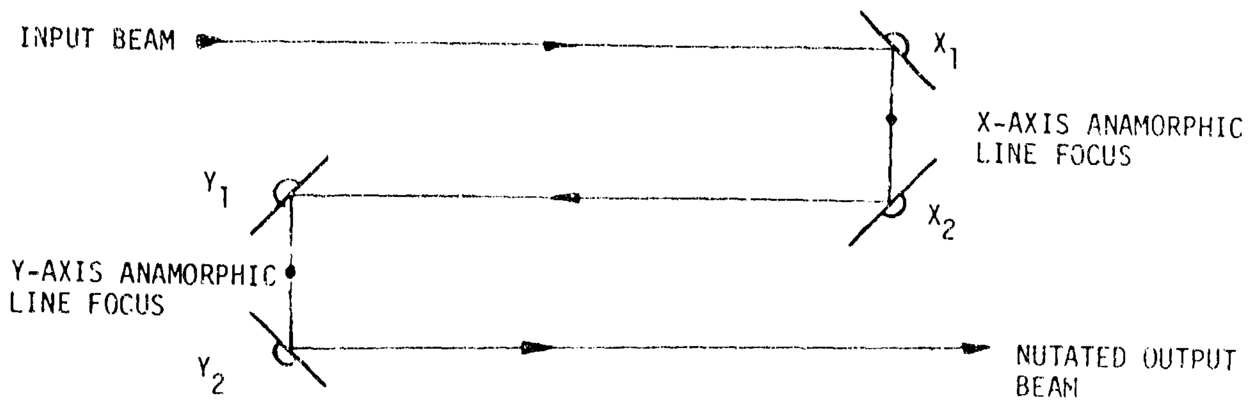
### 6.2.1 Four-mirror nutator

A solution to this problem is shown in Figure 6.4. Here the scan mirror in each axis is replaced by a closely spaced pair of mirrors. The basic idea is to drive each mirror of a one-axis pair with a different constant frequency. The amplitude modulated deflection waveform for that axis is then given by the beat frequency modulation between the two mirror drives. One way to view this method is to consider a drive waveform containing only two Fourier components (instead of many, as with triangle wave modulation). Instead of attempting to force a single mirror to respond to this waveform, the components are separately applied to two mirrors and th superposition obtained optically.

The advantage of this system is that all mirrors can be run at a constant phase and amplitude. The techniques for accomplishing this are already demonstrated.

### 6.2.2 Four-mirror nutation patterns

Two classes of nutation patterns can be obtained using the four-mirror system, referred to as the cosine spiral and the collapsing ellipse. They can be defined by the drive waveforms for each mirror:



Four-mirror nutator geometry.

FIGURE 6.4

Cosine spiral nutation:

$$x_1(t) = D_{max}/2 \left\{ \sin 2\pi \left[ f_N + (f_N/2S) \right] t \right\}$$

$$x_2(t) = D_{max}/2 \left\{ \sin 2\pi \left[ f_N - (f_N/2S) \right] t \right\}$$

$$y_1(t) = D_{max}/2 \left\{ \cos 2\pi \left[ f_N + (f_N/2S) \right] t \right\}$$

$$y_2(t) = D_{max}/2 \left\{ \cos 2\pi \left[ f_N - (f_N/2S) \right] t \right\}$$

$x_1(t)$, $x_2(t)$ = waveform drives for x-axis mirrors

$y_1(t)$, $y_2(t)$ = waveform drives for y-axis mirrors

$D_{max}$ = maximum diameter

$f_N$ = nutation frequency

S = number of spirals per pattern (S/2 in, S/2 out)

The composite x and y scan amplitudes will be (by a simple trigonometric substitution):

$$x_{spiral} = D_{max} \sin 2\pi f_N t \; \cos 2\pi f_{N/S} t$$

$$y_{spiral} = D_{max} \underbrace{\cos 2\pi f_N t}_{\text{circular nutation at } f_N} \; \underbrace{\cos 2\pi f_{N/S} t}_{\text{amplitude modulation at } f_N/S}$$

Collapsing ellipse nutation:

$$x_1(t) = D_{max}/2 \left\{ \sin 2\pi \left[ f_N + (f_N/2S) \right] t \right\}$$

$$x_2(t) = D_{max}/2 \left\{ \sin 2\pi \left[ f_N - (f_N/2S) \right] t \right\}$$

$$y_1(t) = D_{max}/2 \left\{ \sin 2\pi \left[ f_N - (f_N/2S) \right] t \right\}$$

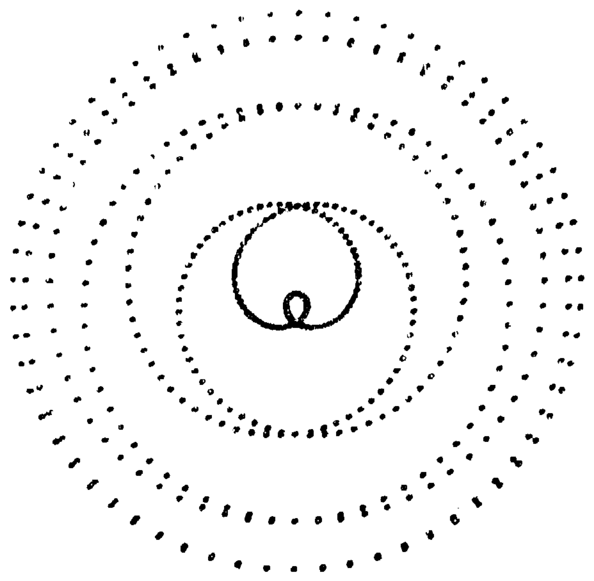
$$y_2(t) = D_{max}/2 \left\{ -\sin 2\pi \left[ f_N - (f_N/2S) \right] t \right\}$$

$$x_{ellipse} = D_{max} \sin 2 f_N t \cos 2 \frac{f_N}{S} t$$

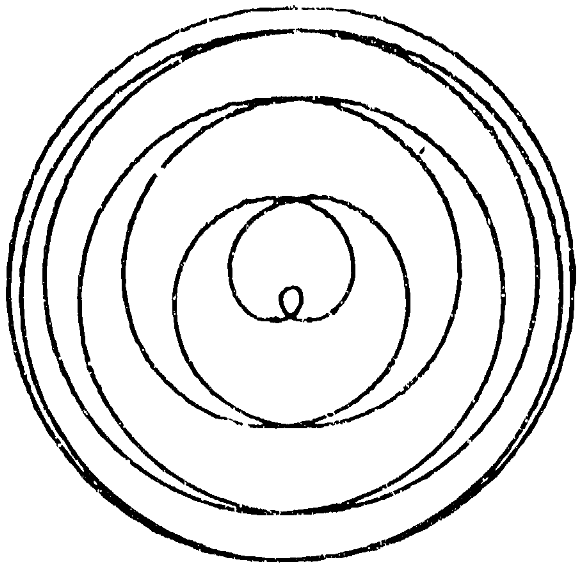
$$y_{ellipse} = D_{max} \cos 2 f_N t \sin 2 \frac{f_N}{S} t$$

Although the cosine spiral and ellipse waveforms differ only in the phase of the y-axis envelopes, the resulting patterns are qualitatively very different.

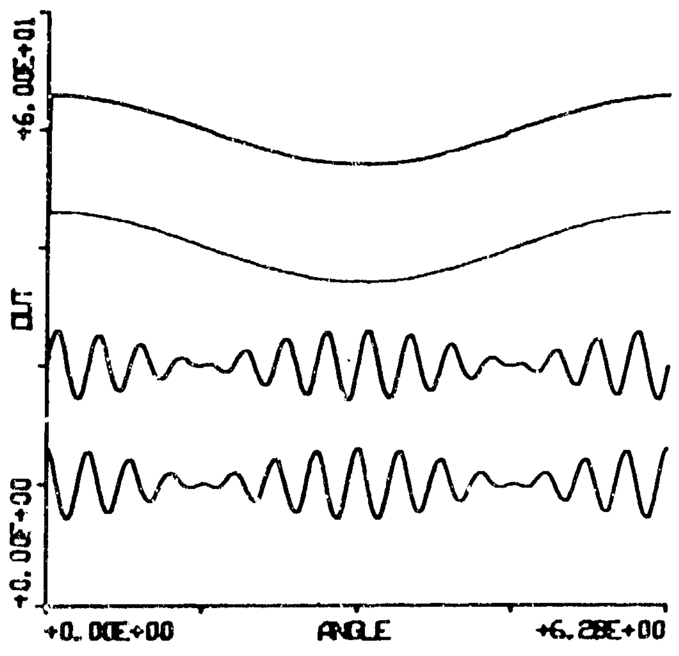
Figure 6.5a shows the sampling density for a cosine spiral with S = 15 revolutions per pattern and 31 time samples per revolution. The nutation path is that shown in Figure 6.5b while 6.5c gives the x-and y- envelope functions and drive waveforms. Although the edge sample density is better than for the linear spiral of Figure 6.3a, the sampling at intermediate radius is somewhat sparse, while the central pixels are still over-sampled. The cosine spiral would still have performance comparable to that of the linear spiral, and is realizable. It remains a poor match for a square reconstruction field, and so the following pattern, the collapsing ellipse, was used instead.



a.



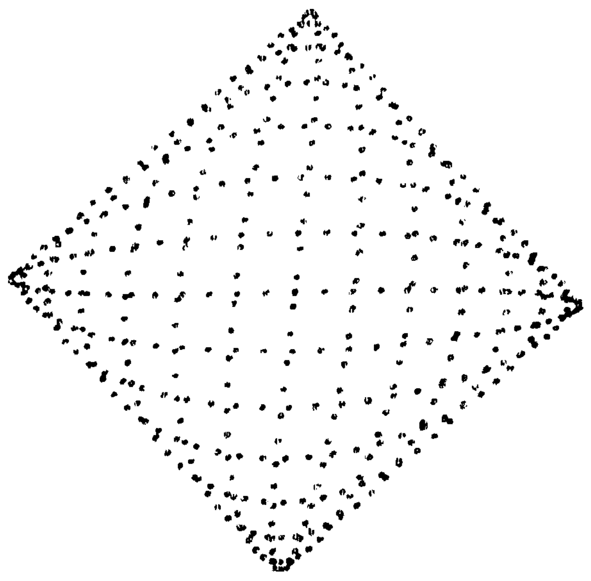
b.

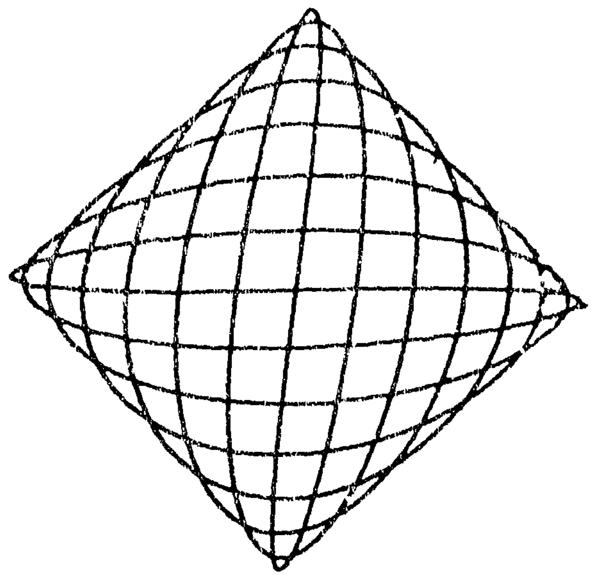


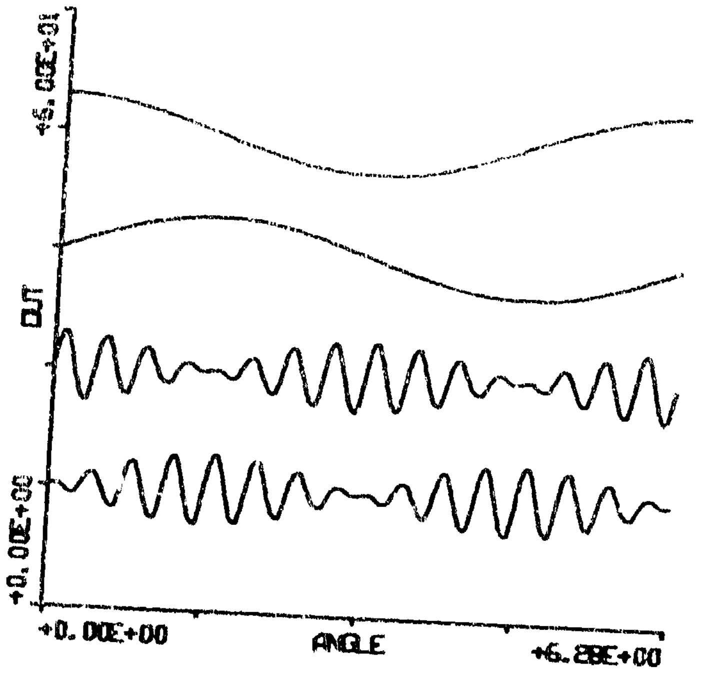
c.

FIGURE 6.5

Figure 6.6a shows the sampling pattern for the collapsing ellipse, for S = 15 revolutions per pattern and 31 samples per revolution. The square fill shape, typical of Lissajous figures, is an ideal match to a square pixel field, and the sampling density is now weighted toward the field edges. Figure 6.7 matches this sampling pattern to an 8 x 8 pixel field. It is seen that the rule-of-thumb of four distinct samples per pixel is obtained for every pixel in the field. Thus, the collapsing ellipse with these scan parameters yields a nearly ideal nutation pattern for image reconstruction. Inspection of the individual mirror drive waveforms shows also that the pattern is generated using only sine components, eliminating the need to maintain a quadrature phase reference, thus simplifying the nutation controller. A set of galvanometers was designed and constructed which could operate at a basic nutation rate of $f_n$ = 7980 Hz.


a.


b.



c.

FIGURE 6.6

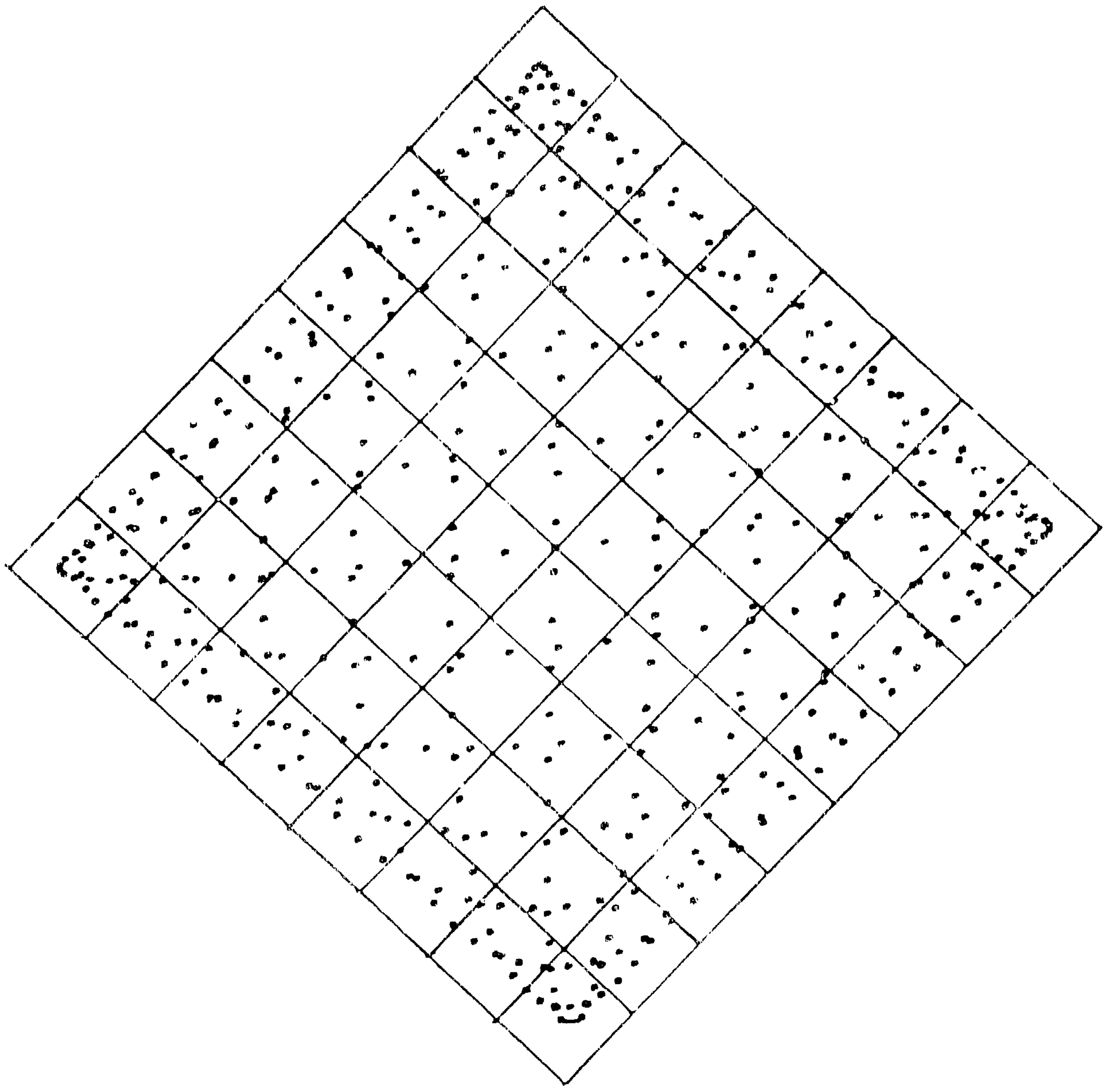


FIGURE 6 7

## 6.3 Interface Electronics

In order to test the imaging capabilities of the AIT breadboard, a set of special interface electronics was constructed to allow the breadboard to operate directly with the DG Nova computer. This enabled development and debugging of the optical subsystems in parallel with work on the PMP.

The interface system is diagrammed in Figure 6.8. The four silicon quad cell outputs are buffered by a set of preamplifiers, and the analog waveforms sent to a set of analog switched integrators. The integrators performed the front end signal integration and sampling required by the AIT algorithm, under the control of timing signals from the DG Nova. The output sampled waveforms were digitized by the Nova DG/DAC and recorded on disk.

In addition to data conditioning, the interface package controlled the AIT slow-speed galvo set. For debugging and experimentation without the PMP, a linear response pair of galvos, with sufficiently wideband, albeit slow, response was used, since the Nova DG/DAC was not fast enough to accommodate the 7980 Hz rate of the fast 4 mirror galvo set. The slow galvos were driven by the compound waveforms of Figure 6.6c, bottom. The waveforms were generated on an Apple computer and downloaded to the interface wave-form generator through an RS-232 interface. The waveforms were ap-plied to a set of driver amplifiers which controlled the linear galvos.

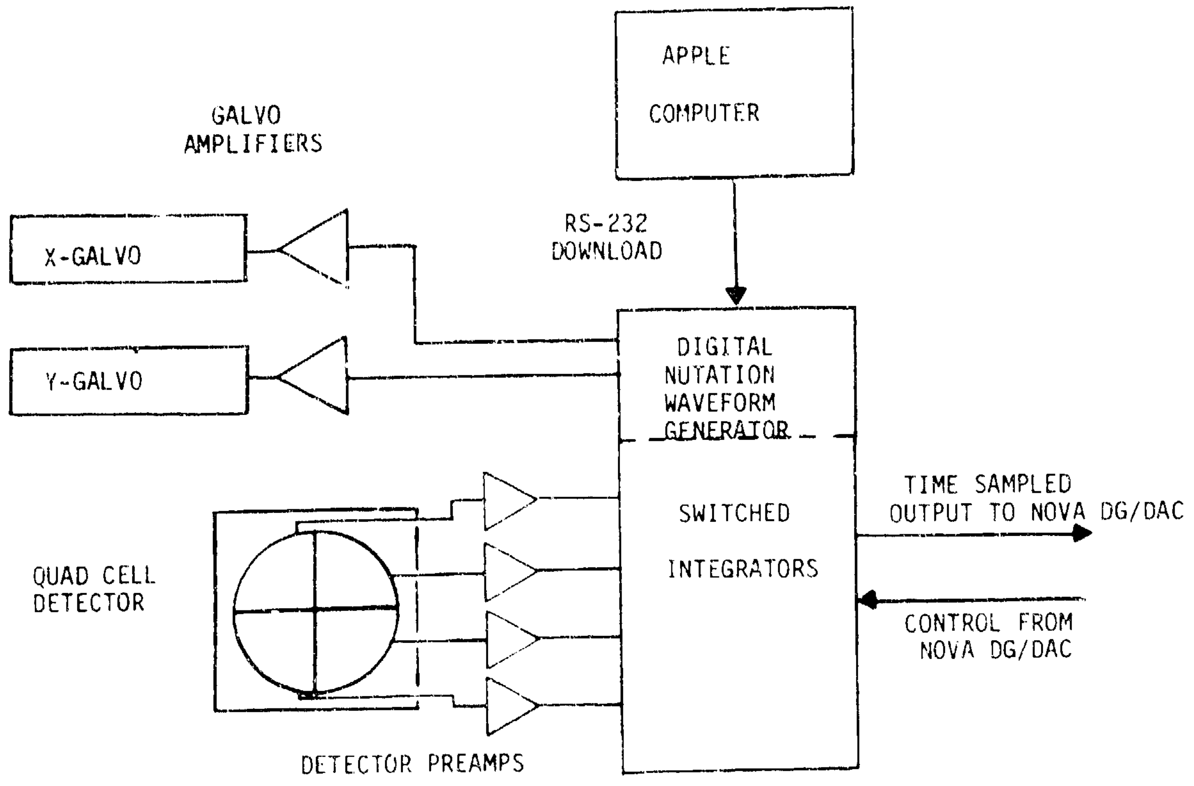

APPLE
COMPUTER
GALVO
AMPLIFIERS
RS-232
DOWNLOAD
X-GALVO
Y-GALVO
DIGITAL
NUTATION
WAVEFORM
GENERATOR
SWITCHED
INTEGRATORS
TIME SAMPLED
OUTPUT TO NOVA DG/DAC
CONTROL FROM
NOVA DG/DAC
QUAD CELL
DETECTOR
DETECTOR PREAMPS

FIGURE 6.8

## 7.0 BREADBOARD HARDWARE: ELECTRONIC PROCESSOR

The complex AIT imaging and tracking algorithms require advanced, highly specialized digital hardware to realize real-time operation. To this end, a custom digital signal processor, the Programmable Microcoded Processor (PMP), was designed and built.

### 7.1 Processor Requirements

The processor unit for the AIT system must be capable of performing imaging, tracking, nutation control, system control, data output and other tasks in real time. This involves accessing of detector outputs, and sampling, digitizing and buffering and organizing the results into two data sets: the major cycle data vectors which are 1860 samples long and must be processed for image reconstruction, and the minor cycle sub-vectors 124 samples long, which must be converted to tracker outputs. Tracking must ultimately occur at 3 kHz update rate, while the imaging goal is 30 frames/sec.

### 7.2 PMP Architecture

The very high data throughput rate required of the system processor precludes use of any form of standard minicomputer architecture. These general purpose machines require decoding of each instruction into the hardware commands; the decoding step takes several machine cycles, greatly diminishing speed. Even advanced array processors are encumbered by use of high-accuracy, floating point arithmetic unneeded here, as well as cumbersome I/O structures, again reducing throughput. The alternative is a dedicated architecture with no instruction decoding, fixed-point arithmetic and optimized

I/O, which meets performance requirements, and is also at least a factor of ten less expensive than the closest commercially available processor configuration.

The PMP is a purely microcode driven digital signal processor with 16 bit data and address busses and 16 bit basic fixed point calculation accuracy. Based on the AMD 2903 family of bit-slice components, the system has a 100 nS cycle time and employs a 96 bit fully horizontal microcode word size.

A simplified diagram of the PMP architecture is shown in Figure 7.1. It is apparent that the system is complex; the speed and flexibility of the processor have been obtained at the expense of a complicated architecture which is challenging to program. The effort has nevertheless been highly successful, resulting in a very reliable system without a single analog adjustment.

The PMP can be approached in terms of functional sub-units or processing resources. The extensive use of tri-state bus switches allows different groups of hardware components to function quasi-independently on different tasks, with the boundaries between groups and their communication channels variable dynamically under micro-program control.

At center left in the diagram are the Am 2903 and 2902 units which form the system ALU. They are loaded and operated by the component group at top center, the Am2904 and Am2910, the 96 bit wide microcode memory and the microcode pipeline register. These devices form a powerful control unit for directing data flow, memory access, external device operation and internal computation with a significant degree of parallelism. The efficient control of

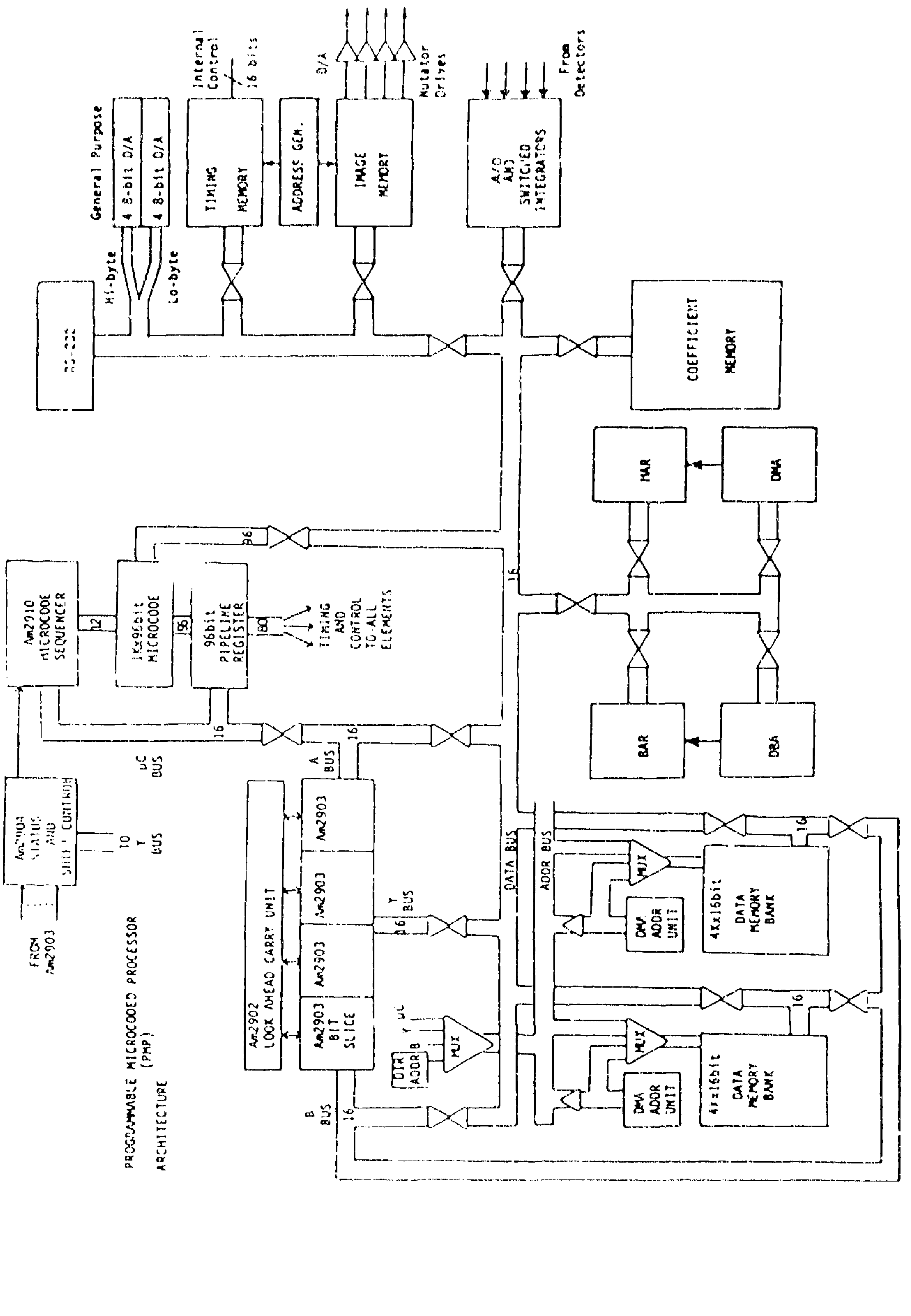

PROGRAMMABLE MICROCODED PROCESSOR (PMP) ARCHITECTURE
RS-232
Hi-byte
Lo-byte
General Purpose
4 8-bit D/A
4 8-bit D/A
TIMING MEMORY
Internal Control
16 bits
ADDRESS GEN.
IMAGE MEMORY
D/A
Nutator Drives
A/D AND SWITCHED INTEGRATORS
From Detectors
COEFFICIENT MEMORY
MAR
DMA
BAR
DBA
Am2910 MICROCODE SEQUENCER
1Kx96bit MICROCODE
96bit PIPELINE REGISTER
TIMING AND CONTROL TO ALL ELEMENTS
12
96
180
16
DC BUS
A BUS
Y BUS
B BUS
Am2904 STATUS AND SHIFT CONTROL
10 Y BUS
FROM Am2903
Am2902 LOOK AHEAD CARRY UNIT
Am2903 BIT SLICE
Am2903
Am2903
Am2903
MUX
DIR ADDR
DATA BUS
ADDR BUS
DMA ADDR UNIT
4Kx16bit DATA MEMORY BANK
DMA ADDR UNIT
4Kx16bit DATA MEMORY BANK

FIGURE 7.1

computational flow is enhanced by the group of four memory units shown below the Am2910 group: the Bus Address Register (BAR) and its Direct Bus Access (DBA) unit, and the Memory Address REgister (MAR) and Direct Memory Access unit. These memories contain the maps and tables needed to fetch and route data in the order required from the main data memory without use of any time-consuming address calculations.

The main data memories are located at lower left. They consist of a pair of 4K by 16 bit blocks with separate access ports and address controls. This allows one bank to input data from some external source (such as the digitized output of a detector set) while the other is read out for processing, thus implementing a "ping-pong" buffer.

In addition to these major PMP hardware units, the AIT version of the PMP contains several other sub-systems, shown at right in the figure. The coefficient memory contains 128K by 16 bit words of storage to accommodate the back matrix for image reconstruction. Inputs from the detector preamps are integrated and digitized in the discrete samples needed by the imaging and tracking algorithms in a detector A/D front end with digitally controlled AGC, shown at lower right.

Generation of up to four separate nutation drives and timing of all nutation-synchronized operations is done by the Timing and Image memories, located at right-center. Analog outputs of essentially any internal values or results, such as x and y tracker outputs, are facilitated by a bank of eight 8-bit D/A converters.

Finally, communication with a host or supervisory computer must be considered. Since the PMP is a dedicated digital system deliberately containing no high-level instruction capability, operator access, system downloading, and other such tasks must be accomplished through some external general purpose computer. This communication proceeds through a standard RS-232 port on the data bus. Through this port any and all memory locations and registers (which are equivalent in the PMP addressing scheme) can be loaded or accessed; in particular, the 1K by 96 bit microcode memory is downloaded through this port.

Major support of the PMP was done through the Data General Nova, on which all microcode was developed using a specially configured meta-asse[illegible]r. For system diagnostic purposes, Pascal code for an Apple computer was also written which allowed it to act as an intelligent terminal for the PMP. These support systems helped make the challenging task of programming the PMP more manageable and eff cient.

## 7.3 AIT Microprogram Structure

The AIT microcode procedures are divided into a set of modules contained within a simple execution loop. The modules are used to sequentially initialize each of the PMP memories with the algorithm parameters (tracker mask function and offsets, nutation waveforms, etc.) required for a particular AIT configuration. As a first step, a 16-word bootstrap loader is entered into the top of microcode memory, which facilitates loading of the remaining micro-program.

The operation of the AIT microprogram is outlined in Table 7.1, while the flow of program control is diagrammed in Figure 7.2. The action of the microprogram proceeds as follows.

After the PMP has been downloaded with the AIT microcode, the microprogram will enter a loop waiting for a word to be transmitted from the host over the RS-232 line. When the word is received, it will be interpreted as a command by the microporgram, which will branch to a specific module using the command code as an offset into a dispatch table. After the command has been executed, control will return to the wait loop (except for the "RUN" command) where the mircoprogram will wait for the next command. Thus, the AIT micro-program contains no executive (only a central dispatcher and several modules), and is totally driven by the host. This allows the PMP to operate in a very general environment with the PMP run-time configuration dictated by the host to allow running different variations of the same experiment.

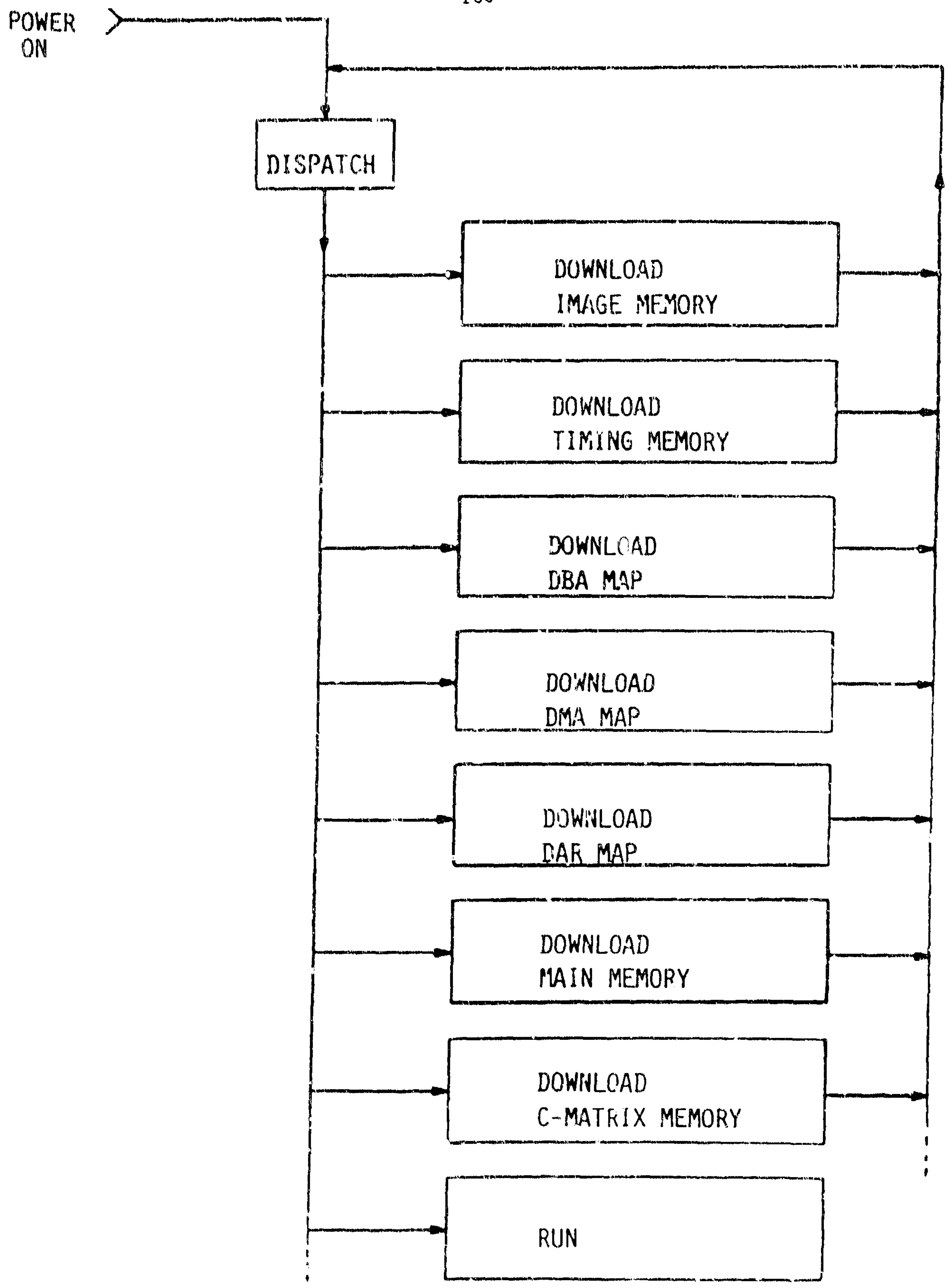


FIGURE 7.2
AIT MICROPROGRAM STRUCTURE

AIT SCENARIO

- DOWNLOAD IMAGE MEMORY. Contains digital codes to be converted to analog signals which drive the two nutation mirrors.
- DOWNLOAD TIMING MEMORY. Contains digital timing and synchronizing information used to integrate and digitize QUAD CELL outputs, and synchronize micro-program to the data acquisition process.
- DOWNLOAD DBA MAP. Contains codes used to address and read the integrating analog to digital converters.
- DOWNLOAD DAR MAP. Contains address pointers used to fetch the digital samples and pre-stored coefficients to allow high speed algorithm execution.
- DOWNLOAD MAIN MEMORY. Contains algorithm constants.
- DOWNLOAD COEFFICIENT MATRIX MEMORY. Contains Imaging Transformation Matrix (Back Matrix).
- RUN.
  1. Initialize and control data acquisition hardware.
  2. Execute Tracking algorithm.
  3. Execute Imaging algorithm.

TABLE 7.1

The PMP contains three autonomous dedicated hardware resources that need to be initialized. One is the Image Memory which contains the digitized nutation control waveforms used to drive the mirrors. The host computer will generate these waveforms and send the sequence of numbers to the PMP, which will, in turn, load the sequence into the memory. In addition, the microprogram will initialize the Image Memory control hardware that is used to determine the nutation frequency. Once initialized, the Image Memory machine is free-running, and the waveform sequence will be repetitively fetched and converted to analog.

The second resource is the Timing Memory which contains timing information used to synchronize all the elements in the system. One bit is used to command the A/Ds to convert and another bit is used to synchronize the microprogram to specific points in the nutation cycle (i.e. tracker cycle endpoints). Again the Timing Memory is loaded by the microprogram with a timing map generated by the host; and its control logic initiated so that these two memories are running in lock step with a programmable phase delay between them. Note that the Timing Memory allows the microprogram to collect and operate on the sampled data in real-time and, thus, optimize the tracking response time.

The third resource is the Direct Bus Access (DBA) machine used to fetch the four integrated-digitized QUAD CELL outputs, under control of a Timing Memory bit. The detector's outputs (at specific points in the nutation cycle, with minimum latency) are stored into a buffer, in a fixed order, to be processed later by the microprogram. As in the previous two memories, this dedicated hardware resource must be programmed by the microcode.

In order to facilitate high-speed algorithm execution, another hardware resource exists called the Direct Access Register (DAR) map. Essentially, this machine maps a count sequence into a random sequence. In other words, a sequence generated by a counter accesses a RAM (containing an address pointer list) which in turn points to a fixed storage location in the main buffer where the data samples are stored; and, thus, obviating any effective address calculations. Also stored in the main data base are various algorithm parameters such as the number of segments in a tracker cycle, gain coefficients, and displacement offsets. Whenever the microprogram needs to access a data sample or a parameter, the microinstruction will issue the micro-orders to access the buffer using the DAR map and advance the DAR counter.

The only unique command is the "RUN" command where the micro-program executes the tracking and imaging algorithms. Note that these two algorithms are independent in that they operate on different data sets and under different constraints, but with some resource sharing. Thus, the environment is similar to multiprogramming with the tracking routine having higher priority.

Essentially the microprogram will periodically test a bit in Timing Memory signifying that all the segments of a Tracker Cycle have been integrated, digitized, and stored in main memory; and ready to be processed. When this condition occurs, the microprogram will branch to the tracking module. The microcode first fetches the parameter that contains the number of segments in the current Tracker Cycle and loads this number into the microsequencer's internal loop counter. Next, using the DAR Map and the loop counter, the segments are accumulated for each of the detector's four quadrants. Then the routine computes the "numerator" and "denominator", performs the division, multiplies the ratio by the gain coefficient, and then adds in the offset. Finally the displacement is checked for overflow and clamped to full scale if necessary.

Whenever the microprogram is not executing the tracking algorithm, and a full frame of "fresh" data has been collected, the microcode will execute the imaging algorithm. Note that main memory is really divided into two separate ping-pong buffers so that where it is time to process a new image frame, the buffers are swung for one major cycle in order to freeze one frame's worth of data. On the next cycle the buffers are swung back so now the imaging data can be accessed without contention over several major cycles.

The imaging algorithm basically consists of computing the inner product of the detector vector (1860 sample points), times the Imaging Transformation Coefficient Matrix (64 x 1860) to yield the pixel vector (64 points). First the detector vector element is fetched from memory (using the DMA address counter) and the coefficient is fetched from the 128K C-matrix memory, then multiplied together (16 x 16 signed multiply), and finally added to the previous product (32-bit accumulation). After the entire column has been multiplied, the pixel value is sent to the host over the RS-232 line. Note that the DC bias must first be removed from the detector vector before multiplication. The DC bias value for each quadrant is specified by choosing those four elements in the detector vector for which it is guaranteed no power fell on the quadrant.

## 7.4 Processor Performance

The PMP was microcoded to perform both the imaging and tracking funcitons and was tested using simulated data sets downloaded from the DG Nova. The processor performed both tasks flawlessly.

As a part of system operation, the PMP had to generate the 15/31 nutation pattern drives for the nutation mirrors. Figure 7.3 shows an x-y oscillogram of the waveform outs of the drives for a two-mirror system. Amplitude modulation of the oscilloscope was controlled by the sample timing pulse. The result is an accurate rendition of the 15/31 collapsing ellipse pattern. This verifies autonomous operation of the Image and Timing memories and associated nutation-synchronized circuitry.

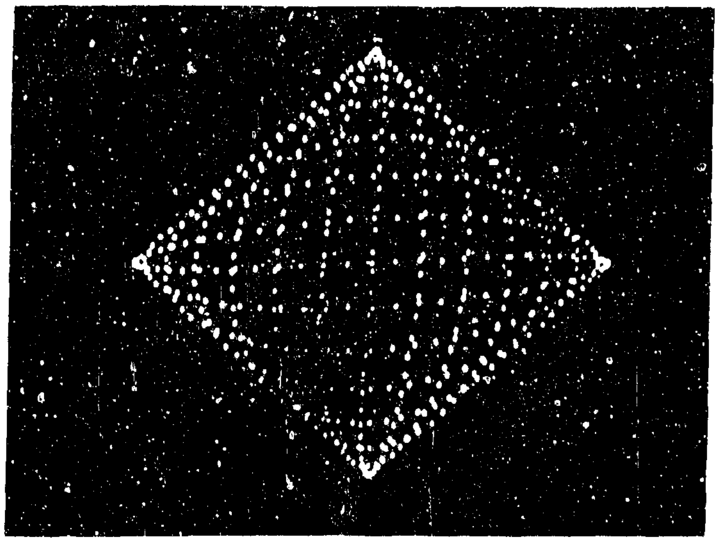
To test the full range of tracker operation, an entire forward matrix was loaded into coefficient memory, where the simulated pixel vectors for each of the 64 pixel locations could be accessed by the PMP in the same way as real-time data. The PMP was programmed to step through each pixel in raster fashion, convert the data to spot location using the AIT tracker algorithm, and output the x,y results through a pair of the general purpose D/A converters.

The separate X and Y analog outputs are shown in Figure 7.4. In 7.4a the full scan is shown, with the upper trace indicating the x-output and the lower the y-output. The expected staircase wave-forms are clearly in evidence, reproducing the pixel locations to an accuracy consistent with the tracker simulation results. Figure 7.4b shows one row of eight pixels (constant y value), revealing the departures from ideal output at the field extremes predicted by the simulations.

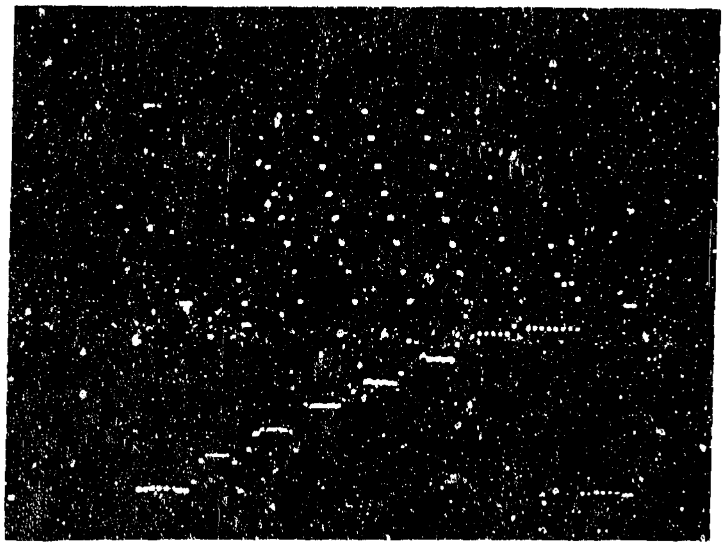
Figure 7.5 shows the results for the entire field, with the x output plotted against y. The grid pattern associated with ideal operation is found in the center of the field, with loss of accuracy occurring in the field corners. Again this matches the predictions of the simulations and shows the excellent performance of both the processor and the algorithm.

To test the imaging microcode, the back matrix was loaded into coefficient memory, and simulated detector vectors were loaded into data memory through the RS-232 port. Two vectors could be loaded simultanecusly, one in each buffer, and alternately processed using the image reconstruction algorithm.

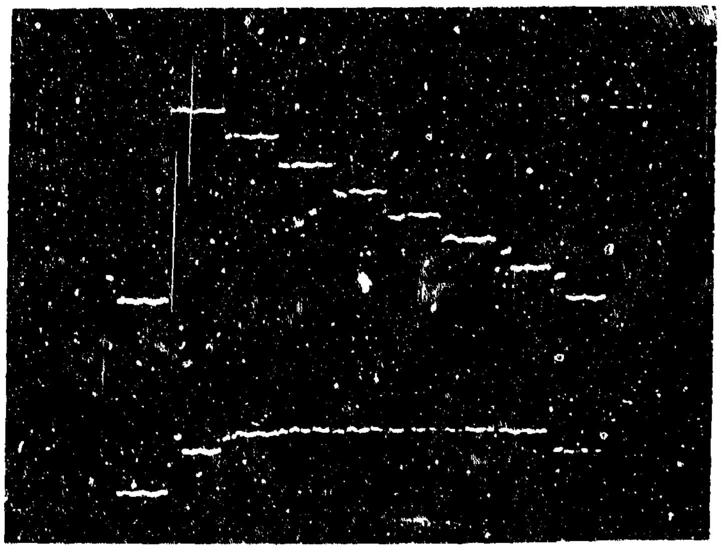
The results were found to be excellent, with reconstruction fidelity using the PMP's 16 bit fixed point arithmetic essentially indistinguishable from the DG Nova floating-point results. Image reconstruction using the PMP alone took approximately 2 seconds; down from the 16 required by the DG Nova with optimized code. Again, perfect operation of the PMP was demonstrated. An increase of frame speed beyond this point requires use of the special hardware multiplier unit, the Vector Matrix Multiplier (VMM).



X-Y PLOT OF TWO
NUTATION DRIVE SIGNALS

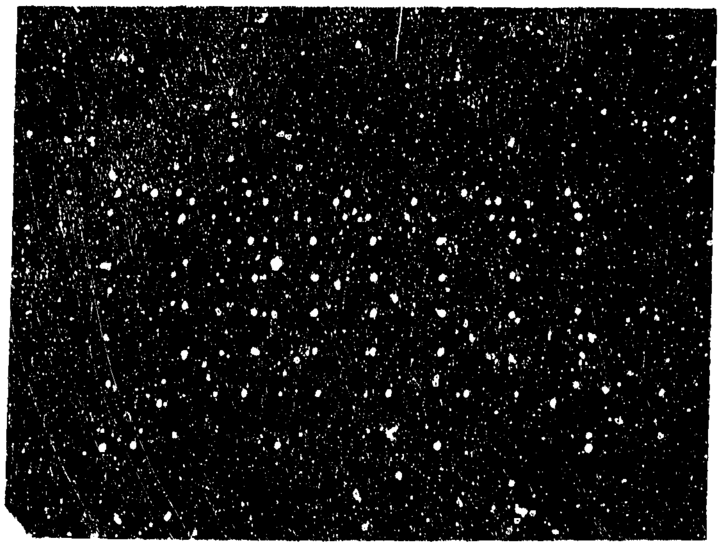


a



TRACKING MIRROR DRIVE SIGNALS

[illegible]



X-Y PLOT OF TWO TRACKING
MIRROR DRIVE SIGNALS

## 7.5 Vector Matrix Multiplier

The Vector Matrix Multiplier, or VMM system, is made up of two units. One unit, called the Vector Processor, consists of a dual vector buffer, a vector matrix multiplier, and the VMM system controller. The other unit, called the Coefficient Memory, consists of an expandable coefficient matrix memory system, as discussed in Section 7.2. Together, these units perform a vector matrix multiplication for a single row vector with a coefficient matrix with the proper dimensions. For the AIT study, the row vector has 1 x 1860 elements and the coefficient matrix has 1860 x 64 elements. After a vector matrix multiplication is performed, a new vector of 64 elements is produced.

### VMM Operating Description

Figure 7.6 describes the VMM architecture. It is divided into a Vector Processor and Coefficient Memory; of which a maximum of eight memory sections or 512K words of memory can be used. Both the Vector Processor and Coefficient Memory have been designed in a pipeline architecture in order to speed the vector processing through-put rate. The maximum through-put rate was designed for 10 MHz, but at the present time the through-put rate is limited to 6 MHz due to the multiplier accumulator device being used (TRW MAC 1010J). As faster multiplier accumulator devices are available, the maximum through-put rate of 10 MHz can be realized.

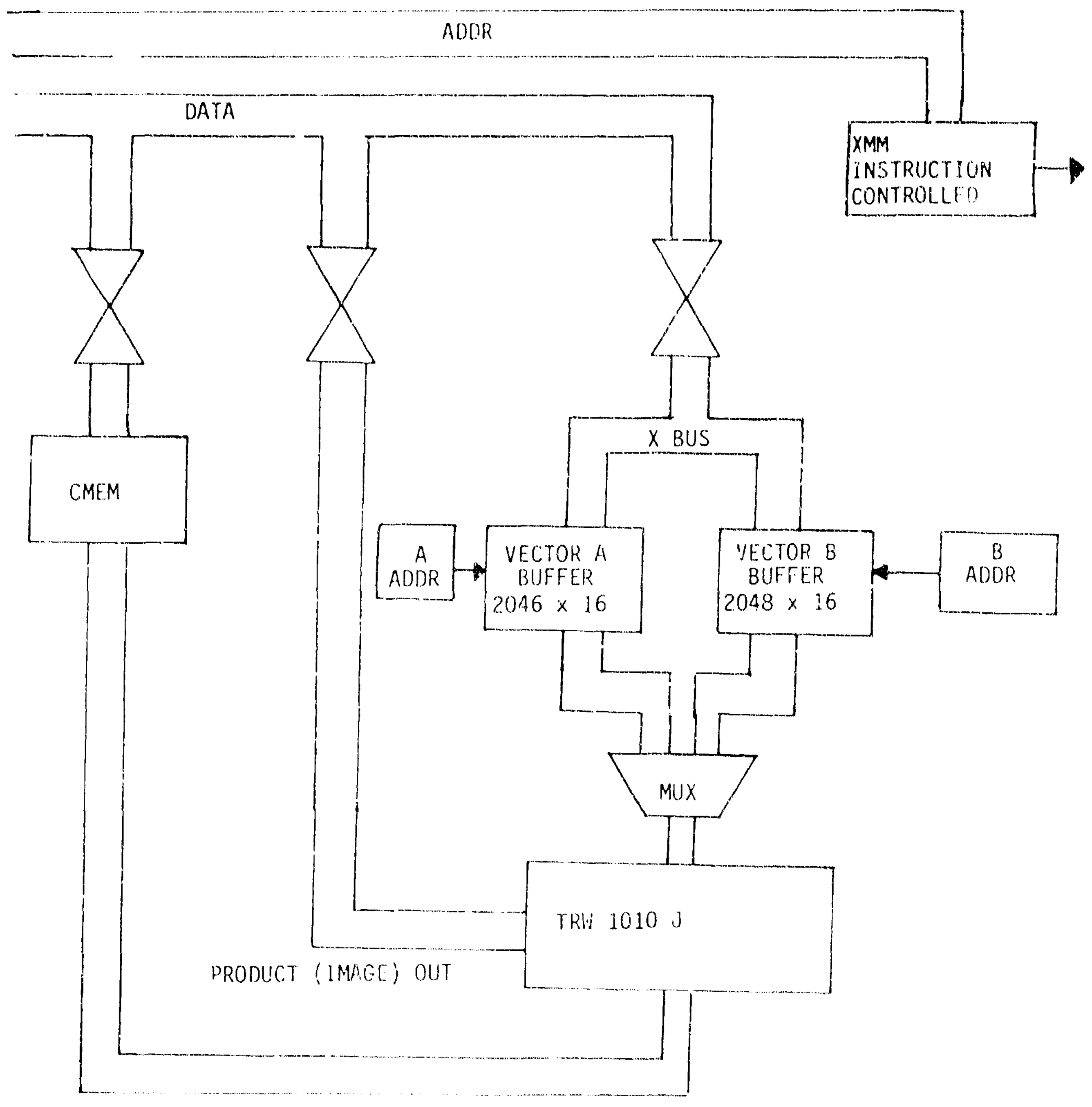

ADDR
DATA
XMM INSTRUCTION CONTROLLED
CMEM
X BUS
A ADDR
VECTOR A BUFFER 2046 x 16
VECTOR B BUFFER 2048 x 16
B ADDR
MUX
TRW 1010 J
PRODUCT (IMAGE) OUT

FIGURE 7.6 VMM ARCHITECTURE

In operation, the VMM vector buffers will accept data from the PMP external bus (x bus) and perform the multiplication and accumulation using the TRW multiplier. The VMM operates under its own clock, so the ping-pong buffering, with the two sections controlled by separate address generators, acts as a synchronous interface between the PMP and VMM. The coefficients are fetched from C-memory via a separate internal bus (in contrast to the way C-memory is used for PMP image processing) to prevent tying up the PMP Data bus by coefficient transfers. The resultant image vectors are sent back to the PMP over the PMP data bus.

The target performance of the VMM is shown in Table 7.2. The hardware is designed to exceed the 30 Hz frame rate goal, allowing a healthy operating margin. Although completely designed and fabricated, the VMM was not debugged and integrated with the PMP by the end of the program, and so full 30 frame/sec processing rate could not be demonstrated.

- Time required to process a vector with one column of the coefficient matrix:

  1860 VMM clock cycles (number elements in vector)
  8 VMM clock cycles (pipeline feed through delay)

  TOTAL 1868 VMM clock cycle (VMMCLK = 6 MHz)
  TOTAL TIME = 312 microseconds

- Time required to read one multiplication process out of the VMM into the PMP:

  10 PMP clock cycle

  The PMP clock period must be 1/2 of the VMM clock period to read the processed result (3 MHz)

  Therefore, the time to read the results and reinitialize another multiplication process is:

  .33 microseconds x 10 clock cycles

  3.3 microseconds/readout

- Total time to process the entire matrix:

  (312 + 3.3) 64 = 20.2 milliseconds

- Conclusion:

  The VMM can process 49.5 vector matrix multiplications per second.

TABLE 7.2 VMM SYSTEM TIMING REQUIREMENT FOR AIT

## 8.0 EXPERIMENTAL SOFTWARE

A large body of software was written to enable the Nova to support all of the experimental data collection, breadboard development, and processor development tasks in the program, apart from the software designed for algorithm simulation. In the sections below, the capabilities of the codes written to support the optical benchwork, PMP development, the Genisco color display, and general system work are outlined. The descriptions are by no means complete, but rather indicate the general intent and emphasis of the software effort. A listing and brief synopsis of all the programs, subroutines and utilities written for the AIT program is given in Appendix I.

## 8.1 PMP Support

Programs were written to provide four types of support to the effort to integrate the AOA-designed programmable microcoded processor (PMP) into the AIT project:

1) Programs to download the "bootstrap" microcode and the microcoded program into the control store of the PMP.

2) Programs to interact with the microcoded diagnostic routines to debug the PMP hardware. There is a "heirarchy" of these program sets. The testing proceded from verifying the integrity of the most vital and basic hardware resources to trying to induce subtle and data-dependent hardware failures.

3) Programs to simulate the microcoded processes using algorithms that mimic these processes. Intermediate results could then be generated and displayed and used as an aid in debugging the microcode.

4) Programs to invoke particular microcode modules and to process and display the results calculated from the data received from the PMP.

All communication between the PMP and the Nova is carried over a standard three-wire RS-232 line. A very simple standard protocol was developed for communication between the PMP and the Nova. This protocol was sufficient for all the modes of PMP support. Under this protocol, the Nova downloads the bootstrap and microcode program to the PMP, indicating which microcode procedure is to be carried out. If more data is required from the Nova for the PMP to carry out this task, more information is sent in this form: a number indicating the number of words in the further communication is sent to the PMP, followed by the data words themselves. Finally, a checksum of all the previous words is sent to the PMP. The PMP, in order to communicate with the slower Nova, awaits a one byte request from the Nova before sending a stream of data (of a well defined type and extent) to the Nova.

This protocol allowed the microcode programs to be built gradually in modules, which could be individually called and tested by the Nova. The modular structure of the PMP code permitted greater efficiency in coding, and also permitted greater flexibility in revising and using the code. Functional microcode modules could be selected from a library of modules and matched to a particular task with minimal overhead.

At the beginning of the effort to achieve PMP-Nova integration, a communications problem was discovered. The Nova RDOS operating system could only recognize numbers corresponding to ASCII "control-S" and "control-Q" as console control characters, even when the user program was employing a Fortran "read binary" command. An assembly language routine using the RDOS system command ".RDS" (read sequential) was written, but these ASCII characters were still not recognized as data. Both input techniques resulted in these characters being intercepted by the operating system and thus, being lost to the calling program.

Faulty RDOS documentation (revealed by a later Data General documentation update) prevented a simple solution to this problem from becoming immediately apparent. A stopgap measure was developed and exploited for the remainder of the AIT project. This solution consisted of writing an assembly language routine to directly interrogate the Universal Line Multiplexor (ULM) board, the hardware intermediary for communications with the PMP and the Dasher printer. This ULM interrogation subroutine was used in all the Nova programs communicating with the PMP. With a simple revision of these programs, ordinary Fortran "read binaries" could now be used. Many of the utilities listed in Appendix I were written to diagnose and correct this problem.

## 8.2 Bench Support

Support for experimental data collection was provided by programs to control the operation of the electronic devices used to drive the nutation mirrors and to sample the detector outputs in synchrony with the sample intervals required by the tracking and imaging algorithms. The mirrors were driven by the interface electronics package (the rack). In order to drive the mirrors, this device required digital information that defines the frequency and the voltage levels of the signals it sends the mirrors, and a representation of the nutation pattern in a form that can be used to drive each mirror.

The programs written to control this hardware were duplicated in a form that could be used by an Apple computer equipped with a digital to analog converter. (These programs were written in "Basic" and in assembly language for the 6502 microprocessor.) The Nova computer was thereby freed from this chore.

The AIT rack also contains switched integrators and sample and hold registers used in the taking of data. These hardware generated timing signals were used by the Nova software to determine when to read the sample and hold registers.

The program EXPDSK contains the code necessary to take data from the sample and holds, scale the data, and store the data to a disk file. EXPDSK incorporates the same header and file name creation subroutines as SIMDSK. The files it generates have exactly the same format at the SIMDSK files, except for a difference in the letter code in the file name.

In order t simplify data-taking from the experimental apparatus, two programs - variants of EXPDSK and Image, respectively named SNAP and FSIMAGE - were created. These two programs are linked to one another by a Fortran "chain", so that each program invokes the other in a continuous process of taking data and viewing the results. Minor changes of the program flow have been written into these programs to make them more convenient to use in this application.

There were stringent limitations imposed on the rate at which data could be taken. These limitations derived from those of the Data General digital to analog converter hardware. Scanning four channels, with external sample and hold registers provided by the AIT rack, no more than 3500 sample points per second could be accommodated. These produced an upper limit of approximately 5 frames per second on the imaging rate.

## 8.3 The Genisco Color Display

Another major software development project undertaken during AIT Phase II relates to the use of the Genisco Color Processing System. This color processor permits a real time display of the 8 x 8 pixel field calculated by the image processing equipment. Use of this color processor required the development of a Fortran-callable assembly language interface (PGPDR, PGPDM) to the processor and also an assembly language program to load the vendor supplied color processor operating system to the programmable color processor (PGP). Using the programming language provided by the PGP operating system and using the Fortran-callable interface, code was written to create pseudo-color displays of the 8 x 8 pixel image, and other code was written to permit x-y graphical displays to appear on the color monitor. An x-y plotting utility permits a 6-color display of the 4 detector vector waveforms and a measurement grid. Displays of other functions occur in other programs. The fast and easy-to-use graphing and imaging tools developed for this contract will be used again and again in future work.

With regards to the real time display of the 8 x 8 pixel image, the Genisco system was originally chosen for its fast data channel interface capability with Data General equipment, and its ability to produce image frames at high rates. A frame rate of approximately 15 frames per second was accomplished, using the relatively high level PGP operating system executive language provided by Genisco.

Using a lower level Genisco color graphics processing assembly language, further improvements in frame rates can likely be accomplished.

These imaging and graphing subroutines (using the Genisco PGP system) have replaced the terminal and printer graphing routines previously used by Image and SIMDSK. These subroutines and their variants have also been used extensively in other programs.

## 8.4 General Disk File Handling Utilities

A general disk file handling package developed for the AIAO contract was adapted for use in AIT software and was widely used. The package includes routines to write out to disk a parameter header and sets of data and other routines to read back the header and the data sets. There is also a file naming utility and an error handling utility. The naming utility creates a disk file with a unique name based on the source of the data (optical bench, or simulation, or other) and also derived from the date and time of the file creation. The error handler sends error messages to the console and returns control of the program to an appropriate point in the program (that is, it provides an error return as well as a normal return).

These routines were designed with the goal of being made general enough to accommodate data sets of different sizes and structure, for example, with different nutation patterns and sampling schemes. But, they were made modular enough and simple enough to allow easy implementation and adequate standardization for uniformity of use.

## 8.4 General Disk File Handling Utilities

A general disk file handling package developed for the AIAO contract was adapted for use in AIT software and was widely used. The package includes routines to write out to disk a parameter header and sets of data and other routines to read back the header and the data sets. There is also a file naming utility and an error handling utility. The naming utility creates a disk file with a unique name based on the source of the data (optical bench, or simulation, or other) and also derived from the date and time of the file creation. The error handler sends error messages to the console and returns control of the program to an appropriate point in the program (that is, it provides an error return as well as a normal return).

These routines were designed with the goal of being made general enough to accommodate data sets of different sizes and structure, for example, with different nutation patterns and sampling schemes. But, they were made modular enough and simple enough to allow easy implementation and adequate standardization for uniformity of use.

## 9.0 EXPERIMENTAL BREADBOARD RESULTS

The experimental breadboard system was set up for visible wavelength operation and interfaced to the DG Nova through the AI1 Interface Electronics rack and the DG/DAC laboratory interface. A binary intensity bar target mask was installed in the target projector, and the nutation amplitude adjusted so that the bar presented as having dimensions of approximately 2 x 4 pixels in the reconstruction field. Because of the anamorphic field lens, the pattern scanned had a roughly 5:3 aspect ratio resulting in the distorted reconstruction field of Figure 9.1. As a result, the bar would appear to be about 2 x 4 pixels in extent for one orientation, but closer to 1 x 6 pixels when rotated $90^0$.

If required, the aspect ratio can be corrected either by an anamorphic re-imaging system or a simple deviation prism-corrector

The nutation drive was set for a frame rate of .5 Hz. This slow speed was found desirable to minimize the phase shift in the linear galvo response. In this experiment, no direct sensing of the galvo mirror positions was used, either for mirror control or system synchronization. In the high speed system, mirror position is sensed dynamically and used in a feedback loop to stabilize the nutation pattern; here the phase shift pattern does not arise.

Using the SNAP program, data sets were taken from the bench, signal conditioned, and converted to output images. Systematic phase offsets were removed using the shifting algorithms and DC restoration was performed on the experimental waveforms, as described in Section 4.

RECTANGULAR NUTATION PATTERN

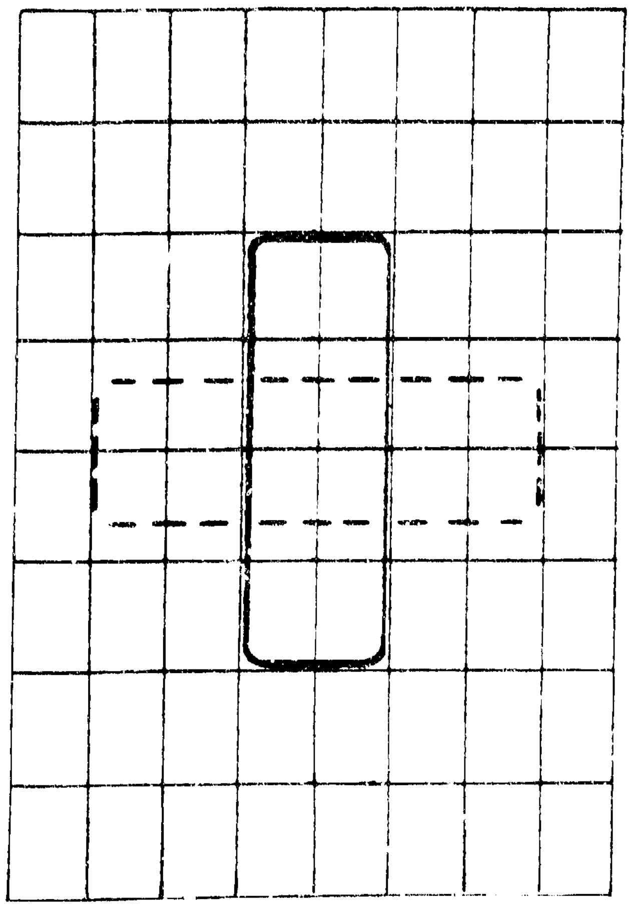


FIGURE 9.1

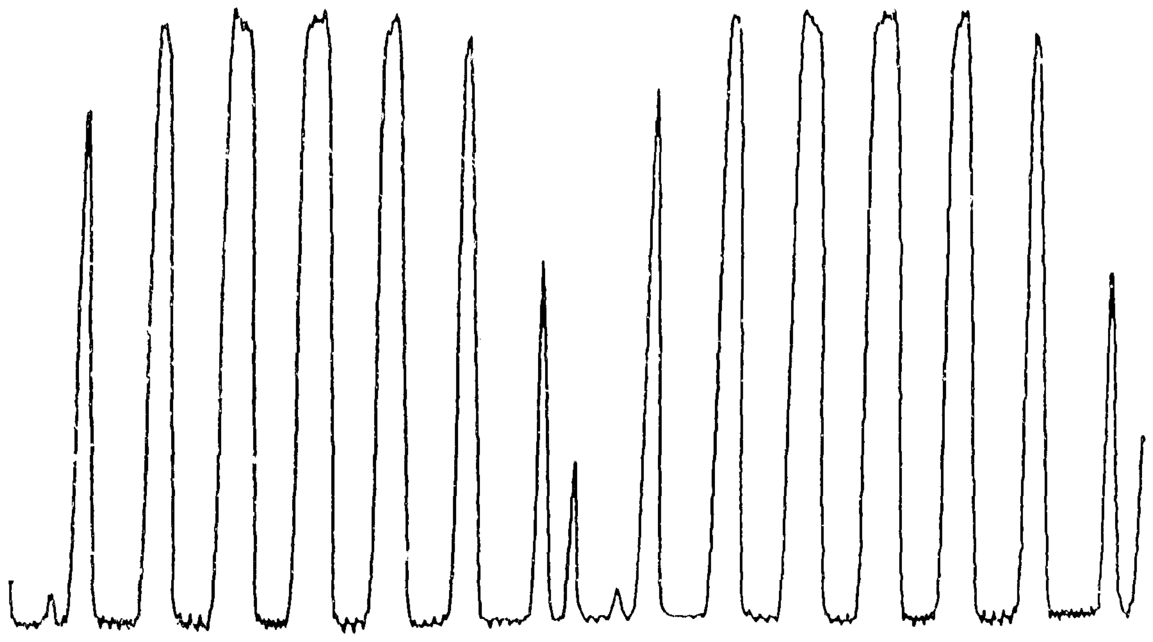
A comparison between the simulated detector waveform for a horizontal bar and the experimental waveform, as measured at the output of the switched integrator, is shown in Figure 9.2. The lower trace is the simulation of quadrant 1, while the upper is quadrant 1 experimental.

The results of reconstruction of the experimental waveform are shown in Figure 9.3. Figure 9.3a is the experimental result, and 9.3b is the simulated reconstruction, reproduced for comparison.

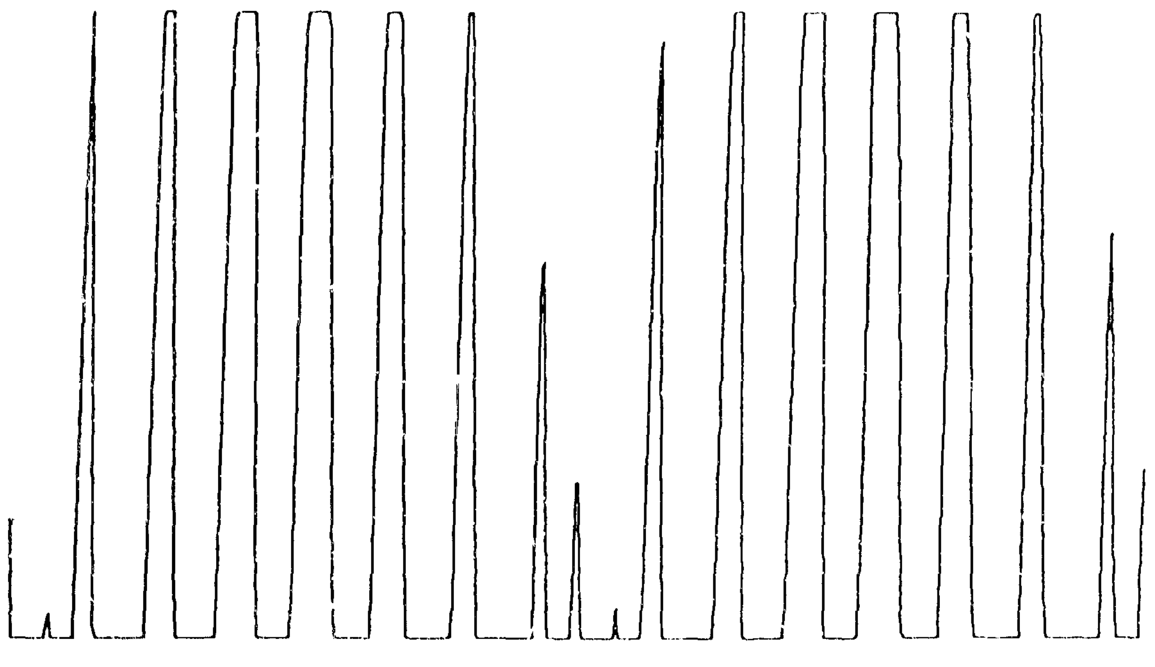
The experimental bar is imperfect but good; much of the reduction in uniformity from simulation is due to the slightly smaller size of the real bar and the fact that the bar is not precisely registered on the reconstruction field. The rendition is accurate, and is the final proof of the efficacy of the entire AIT imaging approach.

The images in Figure 9.4 a - h are taken from a sequence of 32 reconstructions of the bar as it was rotated through $360^{o}$. The display primarily shows the different effects of coarse image sampling. In Figure 9.4 , the end of the bar which is rotating eccentrically actually touches the edge of the reconstruction field; the results of aliasing of outside power are not apparent, indicating that the algorithm is robust with respect to small departures of the image from the reconstruction definition field.

The reconstruction of images from real input data using the AIT breadboard optics was the major goal of the AIT program. It is now possible to carry out a series of thorough experimental studies using the AIT system.

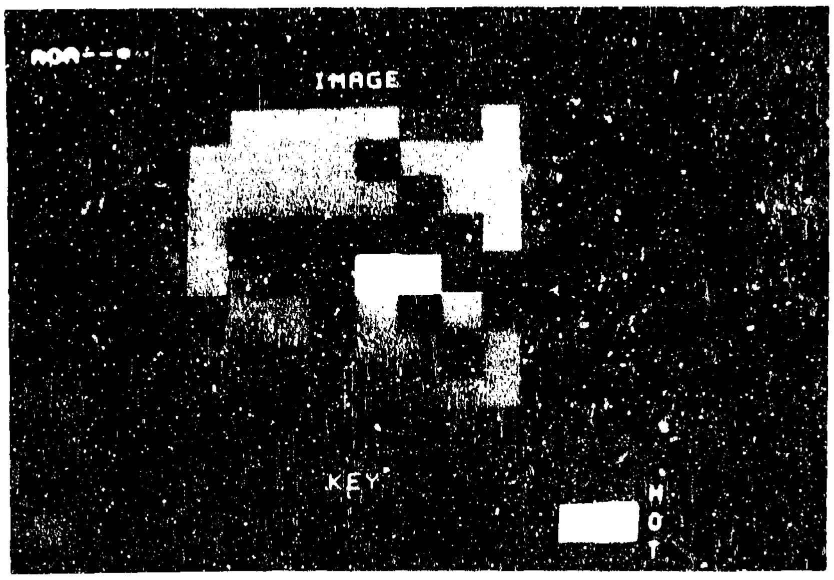


QUADRANT 1: EXPERIMENTAL WAVEFORMS



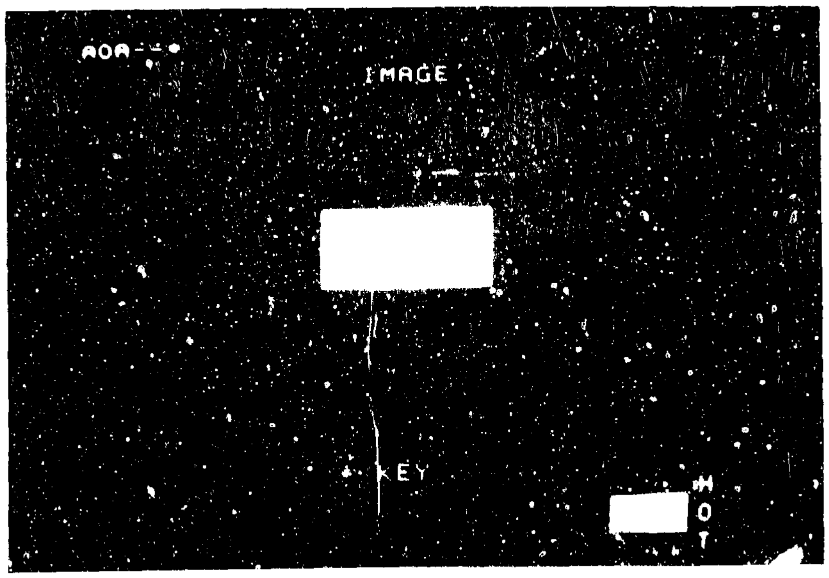
QUADRANT 1: SIMULATED WAVEFORMS

FIGURE 9.2



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 | -10 | -6 | 5 | -13 | 12 | 22 | -4 |
| -6 | .4 | 3 | 5 | 20 | -26 | -6 | 5 |
| 4 | 4 | -8 | -12 | 2 | 19 | -7 | -2 |
| -5 | 10 | 54 | 67 | 48 | 56 | 71 | -6 |
| -12 | 10 | 73 | 81 | 115 | 103 | 83 | 35 |
| 9 | -3 | -5 | 19 | 5 | 31 | -5 | 9 |
| 8 | -13 | -12 | 13 | -9 | -2 | 30 | 4 |
| -11 | 13 | 11 | -5 | 9 | -1 | -31 | 4 |

FIGURE 9.3a



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -4 | 2 | -1 | 0 | 1 | -1 | 2 | -4 |
| 2 | -1 | 0 | 0 | 0 | 0 | -1 | 2 |
| -1 | 0 | -1 | 0 | -1 | 0 | -1 | 0 |
| 1 | -1 | 205 | 206 | 206 | 204 | 0 | 0 |
| 0 | 0 | 204 | 206 | 203 | 205 | -1 | 1 |
| 0 | -1 | 0 | -1 | 0 | -1 | 0 | 1 |
| 2 | -1 | 0 | 0 | 0 | 0 | -1 | 2 |
| -4 | 2 | -1 | 1 | 0 | -1 | 2 | -4 |

FIGURE 9.3b

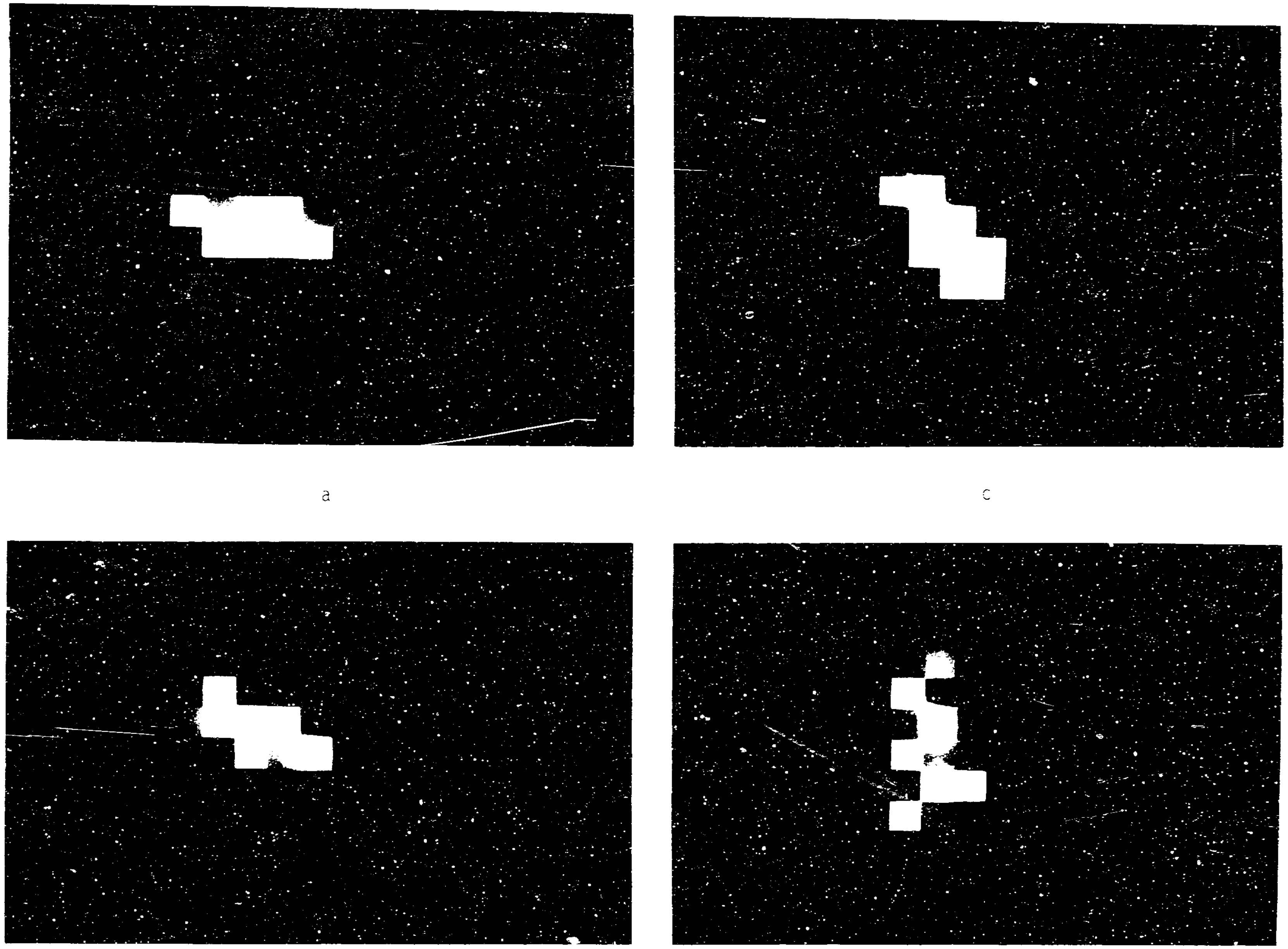
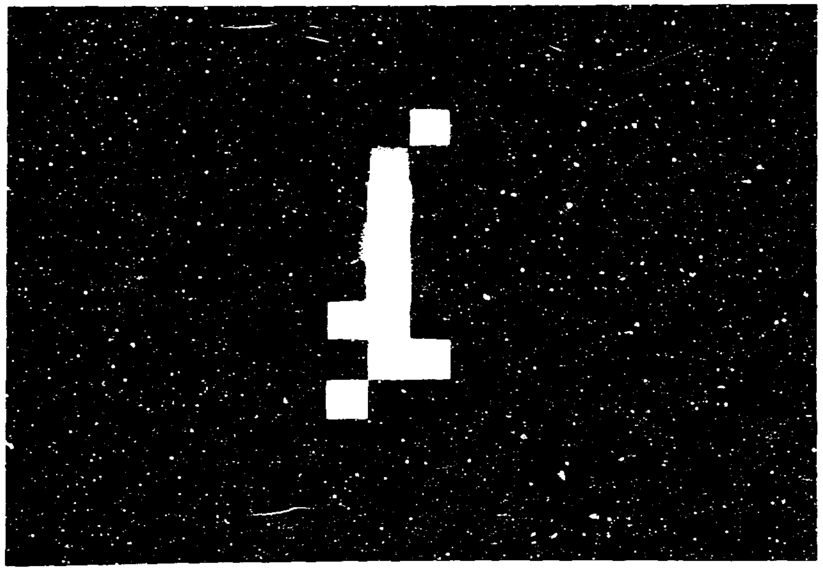
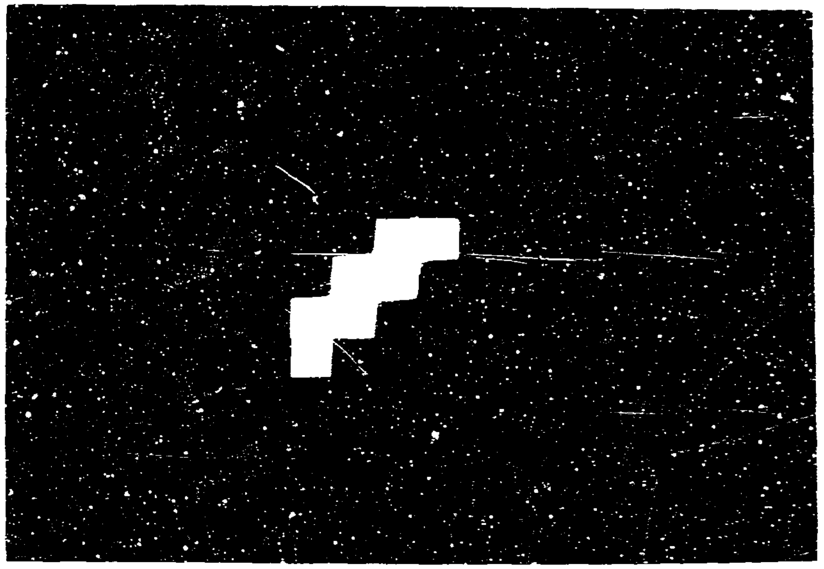


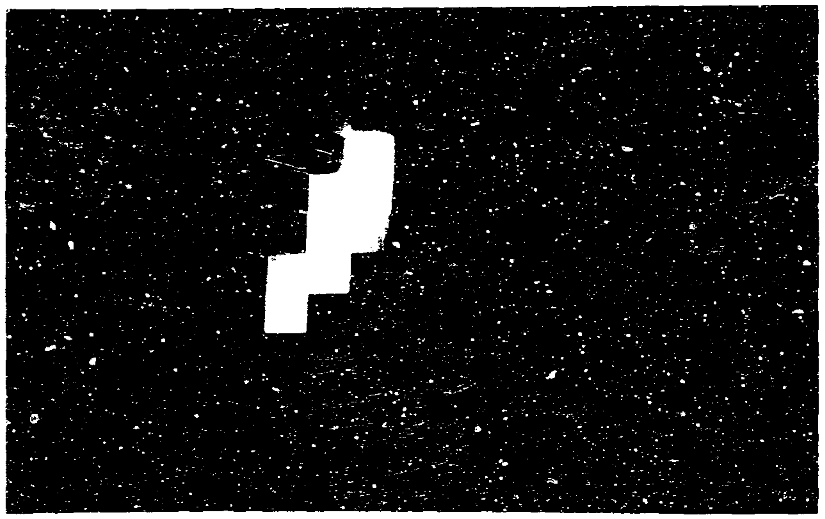
FIGURE 9.4



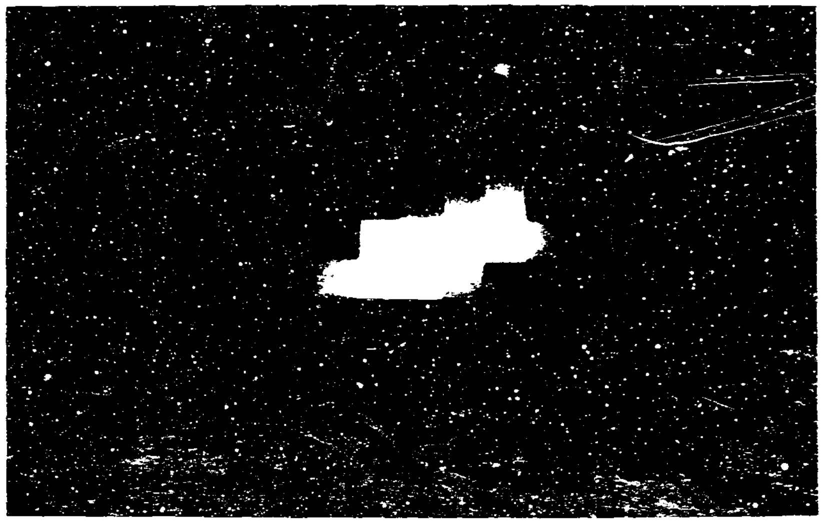
e



g



f



h

FIGURE 9.4 (cont'd)

## 10.0 SUMMARY AND CONCLUSIONS

The AIT program has in total been highly successful. The basic goal of the program has been reached: namely, a new form of imager-tracker in which high-accuracy, high-update rate centroid tracking is obtained simultaneously with medium-speed imaging has been demonstrated. In the process a number of significant theoretical and technological contributions have been made:

1) The AIT class of imaging algorithms has been analyzed and formally characterized. A general method for obtaining image reconstruction algorithms for any resolution has been defined. Requirements for spatial sampling, computational accuracy, and signal conditioning have been established. The accuracy of image reconstruction has been quantitatively examined in simulation. The signal-to-noise characteristics of the imaging process have been defined, and the results compared with other imaging techniques. Particular image-enhancement characteristics of the reconstruction process have been discovered.

2) A class of tracker algorithms suitable for the AIT process has been defined. A prescription for generating an algorithm appropriate to any object size at any resolution has been obtained. The tracker algorithm has been extensively characterized in simulation and found to have excellent properties, displaying good performance far outside its optimal dynamic range. The tracker S/N performance has been estimated and approaches that of an ideal quad-cell tracker.

3) A powerful new digital signal processor, the PMP, has been designed, constructed, and has successfully executed both imaging and tracking algorithms. Tracking computation speed was demonstrated at a full 8 kHz rate, while image rate reached a speed of 2 seconds/frame. A fast vector processing unit, the VMM, was designed and constructed to exceed the 30 Hz frame rate goal, but was not tested.

4) A portable AIT optical breadboard was designed and fabricated, and used to demonstrate reconstruction of real images. The system is capable of visible and infrared operation, and includes a dynamic target simulator with control of target rotation, position, and contrast. The breadboard can operate at variable low nutation rates with the DG Nova computer or at fixed high rate with the PMP. A special 4-mirror scanner technique was invented and its properties formally characterized, so that an appropriate nutation pattern can be defined for any resolution.

5) A large body of experimental software was written, including codes for simulation of tracking and imaging, experimental data collection, PMP development, and color display support. The software is in the form of many utility routines called by a few large programs, allowing flexibility in configuring the programs for new tasks. The AIT code development forms a substantial basis for conducting new investigations into related forms of imaging and signal processing.

With the AIT breadboard system complete, experimental characterization of the imaging and tracking algorithms may be carried out. This would be of particular interest in the infrared mode of operation. Other types of detectors can be employed, since the algorithm formulations are quite general and not limited to use with the quad cell detector configuration. The image processing properties of the technique can also be explored since the appropriate software and hardware tools are in place.

The AIT program output has met or exceeded most of its goals and expectations, and can likely form a nucleus for significant further developments.

## REFERENCES

1. U.S. Patent No. 4,141,652

   L.E. Schmutz, J.K. Bowker, J. Feinleib, S. Tubbs, "Integrated Imaging Irradiance ($I^3$) Sensor: A New Method for Real-Time Wavefront Mensuration," Proc. of the SPIE, 179 (1979).

   L.E. Schmutz, J.K. Bowker, J. Feinleib, S.N. Landon, S.J. Tubbs, "Experimental Performance of the $I^3$ Wavefront Sensor for Closed-Loop Adaptive Optics," Proc. of the SPIE, 228 (1980).

   "Advanced Wavefront Sensor Concepts," Final Technical Report, RADC TR-80-368, January 1981.

2. M. Elbaum and P. Diament, Appl. Opt., 16, 2433 (1977).

3. A. Gelb, Ed., Applied Optimal Estimation, (M.I.T. Press, Cambridge, MA, 1974) pp. 23-24.

4. W.K. Pratt, Digital Image Processing, (John Wiley and Sons, New York, 1978) pp. 388-407.

## APPENDIX I: SOFTWARE SUMMARY

This appendix contains a compilation of all the software developed for the AIT program. The documentation is organized approximately along the lines of usage within the AIT program. Main programs are listed first, and all subroutines last. This listing is intended as both an overview of the code and as a guidebook for new users of the system. Further information is available in "help" files located on the disks where the source files are stored, and in the more general support documentation for the Data General and Genisco equipment used.

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| MOST IMPORTANT DATA FILES | BEGEND | | | This file is created by programs AITSCS or GOCALC. It contains the beginnings and endings of the arcs used in the AIT SCS tracking algorithms. It is used by all the tracking algorithm programs (and some others) |
| | BOOTCODE | | | This file contains the bootstrap for the PMP in a form ready to be picked up by programs BOOTLD or MARCH and downloaded to the PMP. BOOTCODE was translated by program BTSTRP from a meta-assembler object file, named BTSTRP.OB. |
| | DIAGNOSTIC | | | This is the original tape file of the Genisco PGP diagnostic test program. See documentation on the creation of the operating system and diagnostic program in Doug's files. |
| | GAINOFF | | | This file is created by programs AITSCS or GOCALC. It contains the gains and offsets of the tracker cycles of the AIT SCS tracking algorithms. It is used by all the tracking algorithm programs (and some others) |
| | G | | | This is the Genisco operating system data file with the missing words from the tape replaced. It is a working version. Also called GENOP. |
| | GENOP | | | This is the Genisco operating system data file with the missing words from the tape replaced. It is a working version. Also called G. |
| | IMAGECODE | | | This is a microcode program in a form that could be downloaded to the PMP. To use it, rename it PROGCODE and use BOOTLD or MARCH. This program is designed to be run with Nova programs PGPMULT or PGPMULT64. |
| | TRA[illegible]CODE | | | This is a microcode program in a form that could be downloaded to the PMP. To use it, rename it PROGCODE and use BOOTLD or MARCH. This program is designed to be run with Nova programs VECTLD or FVECTLD. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| IMAGE, SIMOSK, AND RELATED PROGRAMS | ACGEN | ACFLGN, MUTLPS | | This is a program to find 4 sets of points (one set per quadrant). These are the points radially most distant from each quadrant for the elliptical nutation pattern. It can be instructed to find up to 8 such points per quadrant. These points are used in the AC filtering subrouting ACFILT. |
| | CIMAGE | PRINTW, CILC, [illegible]D, ACFILT, ALG1, ALG2, ACCPL, COND, COMP, IPRINT, PLOTW, PGP, PGPOR, FIND, subroutine in Disk utility library* DSCUT/1.LB | | Program for AIT to read detector waveform disk files and to test various algorithms for image reconstruction. Optional filtering is also available. Printing and plotting of data may be selected. This is the latest and most elaborate version of program IMAGE. This program is basically a menu-driven skeleton that calls various subroutines to do the work. For further documentation, consult these subroutines. |
| COMP.FR | COMP | NONE | | This is extremely simple utility program to compare two files for exact equality. It prints out the character number at which the two files first differ and the integer value of the first word at which they differ. Similar to CLI utility FILCOM. |
| DSTB.FR | DSTB | NONE | | DSTB in its present form is the first program in a circular chain made up of SMDSK and IMGSW. SMDSK and IMGSW are chained versions of SIMDSK and IMAGE. DSTB opens the file DSTRB created by the program DSTRB (see DSTRB), and rewrites the distribution. It rewrites this distribution of the beginning of each chain cycle. Thus, distributions may be created, turned into detector vectors and analysed into images and the appropriate information printed out, in a repeating fashion, without operator intervention. |
| | DSTRBVW | NONE | | This program prompts the user for the name of a "distribution file". This is defined as a file that may be used as an input energy distribution to the program SIMDSK. DSTRBVW displays this distribution on the terminal screen. |
| DSTRB.FR | DS.RB | NONE | | Is first program in a chain of programs consisting of: DSTB, SMDSK, and IMGSW. DSTB, SMDSK and IMGSW form a circular Fortran chain. DSTRB starts the process, but is not in the chain. DSTRB essentially only creates the file DSTRB and initializes two variable for the next program in the chain. A variant of this program DSTRBT exists that may be used to create a user-defined distribution. |
| DSTRBT.FR | DSTRBT | NONE | | DSTRBT may be used to generate distributions for general use. These distributions form input energy distributions for the program SIMDSK. They are 20x20 real arrays. The coordinate system used in this program is a little ridiculous to try to describe. A pictorial representation is available in the AIT written documentation (Note, not in Final Report, but in in-house documentation). The figure is described to the program in terms of blocks, ie. rectangular sections. Each block is determined by inputting 2 x,y coordinate points. The order in which the 2 points are input is important. The first point should be the coordinates of the lower left corner of the block. The second should be those of the upper right corner of the block. The order in which the blocks are presented to the program does not matter. The distribution is output to a file named DSTRB. Program DSTRBVW may be used to view the results |
| | | | | |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| IMAGE, SIMDSK, AND RELATED PROGRAMS (cont) | FRWMATVW | ALG2FM, FMDP PGP, FIND, PGPDR | | A very special purpose and somewhat useful variant of IMAGE. This program opens a file named FMAT (It assumes FMAT contains a 1860x64 matrix), and displays the result of multiplying it by BMAT, the back matrix, on a pixel by pixel basis. It displays the images on the Genisco monitor. |
| | FSIMAGE.FR (with overlay file) | | | Requires all CIMAGE subroutines with GETDAT replaced by FSGETDAT (see program CIMAGE). Program for AIT to read detector waveform disk files and to test various algorithms for image reconstruction. Optional filtering is also available. Printing and plotting data may be selected. This program is the second program in circular Fortran chain. The first program in the chain is SNAP (see program SNAP). This program operates similarly to IMAGE except in the following ways: It does no prompt for name of input file. It always uses file SNAPDATA as input. It automatically performs algorithm 2 (matrix multiplication) on the waveform vector in SNAPDATA and after obtaining the image vector, it automatically displays the image on the Genisco. At this point, it displays the usual image menu prompt and you may proceed as usual. (However, a call to algorithm two will always be automatically followed by a Genisco display.) If FSIMAGE was swapped for SNAP, after an orderly exit from FSIMAGE, control will return to SNAP If FSIMAGE was called as a stand alone program from the CLI, control will be returned to the CLI. |
| | GETPIX | NONE | | Uses file FMAT as a data file. (FMAT is the matrix of basis detector vectors.) Very simple program to obtain a detector vector corresponding to a given pixel and write it out to a disk file. This file is named INVECT. A very useful program. |
| | IMAGE | PRINTW, CTLC, GETDAT, PHEAD, ACFILT, ALG1, ALG2, ACCPL, ALG4, IPRINT, PLOTW, PGP, PGPDR, FIND | | Program for AIT to read detector waveform disk files and to test various algorithms for image reconstruction. Optional filtering is also available. Printing and plotting of data may be selected. This is an older version. CIMAGE is the most recent and most elaborate version. This version is still servicable. |
| | IMGSW | PRINTW, CTLC, GTOTSW.FR, PHEAD, [illegible], ALG1, ALG2, ALG4, IPRTSW.FR PLOTW | | Program for AIT to read detector waveform disk files and to test various algorithms for image reconstruction. Optional filtering is also available. Printing and plotting of data may be selected. This version of IMAGE has been adjusted to be run "automatically" with DSTB, DSTRB, and SMDSK. It uses the waveforms created by SMDSK from the distributions created by DSTB to produce the corresponding images. These programs use the same naming conventions for output files as the standard versions, IMAGE and SIMDSK. |
| | IMGVW | NONE | | Very simple program to view pixel vector intensities, as integer values, on the terminal. The user is prompted for the name of the 128 byte image file. |
| | MATSCAL.FR | NONE | | This program takes a real matrix, 1860x64, and rescales it by multiplying every element by a constant, and fixing the element to be an integer. It prompts user for rescale factor. If user tries to use too large a scale factor, integer overflow (run time error #5) may occur, or, when using the matrix in a matrix operation in another program, a similar error may occur. Therefore, user is responsible for determining appropriate rescale factor. MAXTX or MOTSTAT programs may be used to obtain the maximum and minimum elements in the matrix. File names for input and output matrices must be supplied by user in response to program prompts. Usual two word header code giving row and column dimensions of matrix is assumed by this program. |
| | | | | |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| IMAGE, SIMOSK AND RELATED PROGRAMS (cont) | MATSTAT | NONE | | In its present form, th[illegible] utility program finds the maximum and minimum entries of a double precision matrix file This program also finds mean and standard deviation for the pixel columns on a column by column basis. To use for a real or integer matrix, change "double precision" accordingly, and recompile and reload. The usual two-word header code defining the row and column dime[illegible] of the matrix is assumed by this program. |
| | MAXLPS.FR | NUTLPS | | This is a quick utility program created from ELIPSS.FR to find the maximum and minimum values that the nutation pattern achieves. |
| | MAXTT | NONE | | In its present form, this utility program finds the maximum and minimum entries of a double precision matrix file, dimensiones (1860 x 64) to use for a real or integer matrix, change "double precision" accordingly, and recompile and reload. This program assumes that the matrix file contains the usual 2 word header defining the row and column dimensions of the matrix. |
| PLTWV.FR | PLTWV | LSPLOT, TMPLT, plotter libraries in file DAPLOTS | | Routine to read and plot SIMOSK waveform files on the Hewlett-Packard x-y plotter. P switch forces output to the printer. T switch permits video terminal preview plot K switch supresses plotting of axes on x-y plotter. |
| | PRGHD | HDGN.FR, PHEAD, WRHEAD (IN GSCUTL1.LB) | | PRGHD is a program that produces a file that contains only a header (without waveform data) in the format used by the GETDAT subroutine of IMAGE. It uses program SIMOSK as a prototype. |
| | SIMOSK | HEADGN, NAMFIL, CRIL, WRHEAD, PHEAD, ICUBE, INTEG, WRDATA, GSPSF, GSQSF, INTRO, NLIS, INCIR, INCIG, INOSK, ERRCK, QCGEN, INRI, QVECT, QCELL, GNOIS, RANF, CFLI, NUTLIS, OULAP, ITRAP | | This program simulates the quad cell detector output for a given optical intensity distribution and a given nutation pattern. Integrated data is written to disk for processing by other programs. |
| | SIMRUN | HEADGN, NAMFIL, WRHEAD, PHEAD, ICUBE, INTEG, WRDATA, GSPSF, GSQSF, GSCSF, INTRO, NLIS... (see SIMOSK) | | This program simulates the quad cell detector output for a given optical intensity distribution and a given nutation pattern. Integrated data is written to disk for processing by other programs. This program is a variant of SIMOSK It is used to create sets of detector vector files that will be used by program TRKTRNS. These detector vectors are analyzed in other programs using the AIT SCI tracker algorithm (in program TRKTRNS), and the tracker transfer curves created by this analysis are presented graphically by program TRNSGRPH. The array "name" should be set to appropriate name for the input energy distribution used. |
| | | | | |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| IMAGE, SIMDSK AND RELATED PROGRAMS (cont) | SIMDSK | HDGN.FR, NAMFIL, WRHEAD, PHEAD, ICUBE, INTEG, WRDATA, GSPSF, GSQSF, GSCSF, INTRSW.FR... see SIMDSK for rest of sub-routines | | This program simulates the quad cell detector output for a given optical intensity distribution and a given nutation pattern. Integrated data is written to disk for processing by other programs. This version of SIMDSK has been adjusted to be run "automatically" with DSTB, DSTRB, and IMAGE. It uses the distributions created by DSTB to produce the corresponding waveforms. These programs use the same naming conventions for output files as the standard versions, IMAGE and SIMDSK. |
| | TTR | NONE | | Very simple program to view the contents of a data file containing unformatted (binary write) real numbers. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| BACK MATRIX GENERATION & LINEAR ALGEBRA PROGRAMS | ADDZERO | NONE | | This simple program zeroes out the appropriate detector channels in the forward matrix as are indicated in the body of the code. This is used to generate a one or two detector forward matrix, and use it to generate a similar back matrix. Input matrix is called OLDF.MX, output matrix is called F.MX. |
| | BMATSCAL.FR | | | This is a version of program MATSCAL that rescales the integer matrix BACK.MX generated by DOUGMAC MC. The input and output matrix names are "hardwired" to BACK.MX and BMAT. |
| | CALMTP | MTP | | General purpose routine to call MTP to transpose a matrix. User enters file names for input and output files and gives NBUF size (see comments to MTP). NBUF is allowed to be up to 2048 in size. X switch for double precision; NRI is also allowed to be up to 2048 in size. This means that this program will work for any matrix up to size 2048 by 2048; and if it is dealing with double precision files, it will need to run under SYS4 for that size matrix. Read the introductory comments to MTP for restrictions on the size of NBUF in relation to the matrix size. |
| | DMULT | NONE | | Dedicated DOUGMAC version of a matrix multiplication routine. This program multiplies the transpose of the 64 x 64 inverse matrix, TINV.TX, by the forward matrix transpose, FT.TX. The product is the back matrix transpose, BACK.MX. BACK.MX and FT.TX are single precision, TINV.TX is double precision. |
| | DMULT1 | NONE | | Dedicated DOUGMAC version of a matrix multiplication routine. This program multiplies the single precision forward matrix, F.MX by its single precision transpose (created by this program as a temporary file, FTEMP). The product is a double precision matrix stored in file named FTF.MX. |
| | GBACK | CTLC, MMULT | | Calls MMULT to create back matrix from FTFINV and FT (using the transpose of FTFINV in TINV.MX; MMULT uses transpose of first matrix). Buffer sizes are dictated by MMULT and size of matrices. Compile with X switch for double precision. |
| | GFMX | FORU | | Program to generate F.MX, the forward matrix without doing SIMDSK. Compile with X switch for double precision. |
| | GFTF | CTLC, MMULT | | GFTF generates FTF with a call to MMULT. (FTF is the product of the transpose of the forward matrix and the forward matrix). Matrices are stored on disk with dimensions (rows then cols) in the first 2 16-bit words followed by real or double precision data (row index varying fastest). NUTXY and F.MX must be created before this program can be run. If F.MX is generated as single precision but double precision is wanted from that point, use SDGFTF version and link with SDMULT not MMULT. SDMULT takes single precision input and creates a double precision product. MMULT treats everything as either all single or all double. NR and NC (# rows and cols) are read from F.MX. NC will be NPIX x NPIX, NR will be NOUTxNCYCx4 (total outputs x 4 detectors). Channels: ICHF for F.MX, the forward matrix; ICHFTF for FTF.MX, product of FT.TX and F.MX; ICHTMP is scratch channel used for extra copy of F.MX (TEMPF.MX). Channel save arrays are named similarly (eg. ISVF); files - .MX are matrices, .TX are transposes. Buffers BUF1, BUF2, and BUF3 are dimensioned according to the size of the buffers needed for MMULT. Generally, the dimensions of BUF1 and BUF2 should equal the number of rows in F.MX. BUF3's dimension should equal the number of columns in F.MX. There are two versions of GFTF, the normal one (with an X switch for double precision), which should be used for all single or all double precision calculation; and SDGFTF, which should be linked with the SDMULT version of MMULT for use when the input is a single precision F.MX and the output is a double precision FTF.MX. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| BACK MATRIX GENERATION & LINEAR ALGEBRA PROGRAMS (Cont) | ITEST | MUTLPS, X-Y plotter routines in XY plotter library | | This is a testing version of ILIPS, the routine to generate the elliptical nutation pattern. It is set up to plot the pattern on the plotter. |
| | LOK64 | NONE | | Program LOK64, to look at any 64x64 matrix. Switch X for double precision. Terminal input of choice of output device as well as file name to be examined. Program assumes that the disk file starts with 2 16-bit integer words (dimensions, row first) followed by data (row index varying fastest). |
| | MFINV | CTLC, MTP, MMULT | | Program MFINV - to look at FTF X FTFINV and write this product for the purpose of checking to see whether it is an identity matrix. Compile with the x switch for double precision. |
| | MOVER | NONE | | This program is used to break a large file (for example a matrix file) into a set of smaller files that can be conveniently stored to floppy disks. |
| | NOISE | NONE | | Program NOISE to calculate sum of squares of coefficients of back matrix and noise per pixel - using (VT.TX for easy access to rows - which are columns in BT.TX) - Janie 6/81 Dimension of BUF = number of columns in back matrix (rows in BT.TX); dimensions of other arrays = number of rows in back matrix (columns in BT.TX). Variables:<br>SQTOT: Sum of squares all coefficients<br>PIXSQ(i): Sum of squares, row I of back matrix (Pixel I)<br>CFTOT: Sum of all coefficients<br>PIXTOT(I): Sum of coefficients, row I<br>PXROOT(I): Sqrt of PIXSQ(I)<br>ROOTSQ: Square root of sum of PXROOTS(I) |
| | PDINV | MINV | | Program PDINV inverts double precision matrix, FTF.MX |
| | PIXELS | NONE | | Program to access the forward matrix across sets of 4 rows - ie. looking at all pixels for 1 output step - presumes (at the moment - 6/5/81) 15 nutation cycles, 31 steps per cycle for a forward matrix 1860 rows long. First dimension of pixel array should be number of pixels in the image. This should also correspond to NC, number of columns in F.MX. |
| | SBACK | CTLC | | This program is used to convert BACK.MX to single precision. Dimension of SBACK and OBACK should be chosen according to the size of BACK.MX. The dimension should be a multiple of the number of rows in SBACK (so that the program reads blocks composed of full columns). The quotient of the dimension and the number of rows should be a factor of the number of full columns in the back matrix (so that the program can write blocks of full columns). NFRAC is that quotient (set in a data statement). (For example, for a matrix 64 x 1860, the dimension might be 1280 with NFRAC=20. For a matrix 64 x 2048 the dimension might be 1024 with NFRAC=16). |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| BACK MATRIX GENERATION & LINEAR ALGEBRA PROGRAMS (cont) | GFTF (SDGFTF version) | CTLC, MMULT | | GFTF generates FTF with a call to MMULT. (FTF is the product of the transpose of the forward matrix and the forward matrix). Matrices are stored on disk with dimensions (rows then cols) in the first 2 16-bit words followed by real or double precision data (row index varying fastest). NUTXY and F.MX must be created before this program can be run. If F.MX is generated as single precision but double precision is wanted from that point, use SDGFTF version and link with SDMULT not MMULT. SDMULT takes single precision input and creates a double precision product. MMULT treats everything as either all single or all double. NR and NC (# rows & cols) are read from F.MX. NC will be NPIX x NPIX. NR will be NOUT x NCYC x 4 (total outputs x 4 detectors).<br>Channels:<br>ICHF for F.MX, the forward matrix<br>ICHFTF for FTF.MX, product of FT.TX and F.MX<br>ICHTMP is scratch channel used for extra copy of F.MX (TEMPF.MX)<br>Channel save arrays are named similarly (eg. ISVF); files - .MS are matrices, .TX are transposes. Buffers BUF1, BUF2 and BUF3 are dimensioned according to the size of the buffers needed for MMULT. Generally, the dimensions of BUF1 and BUF2 should equal the number of rows in F.MX. BUF3's dimension should equal the number of columns in F.MX. There are two versions of GFTF - the normal one (with an x switch for double precision), which should be used for all single or all double precision calculation; and SDGFTF, which should be linked with the SDMULT version of MMULT - for use when the input is a single precision F.MX and the output wanted is a double precision FTF.MX. |
| | TBACK | CTLC, MTP | | Routine TBACK to call MTP to transpose BACK.MX (single precision version) to BT.TX - See MTP for comments on buffer sizes and value of NBUF. |
| | TRANS1 | NONE | | Dedicated DOUGMAC version of a matrix transposition routine. This program produces the single precision transposition (in file FT.TX) of the single precision matrix in file F.MX. |
| | TRANSP | CTLC, MTP | | This program is used to transpose matrix in file FTFINV.MX. The resulting matrix is stored in file TINV.TX. |
| | MACRO DOUGMAC<br>RLDR ADDZERO<br>FORT.LB, ADDZERO,<br>DELETE/V ADDZERO.SV,<br>RLDR TRANS1 FORT.LB,<br>TRANS1,<br>DELETE/V TRANS1.SV,<br>RLDR DMULT1 FORT.LB<br>DMULT1<br>DELETE/V DMULT1.SV FTEMP F.MX<br>RLDR PDINV MINV FORT.LB<br>PDINV<br>DELETE/V PDINV.SV<br>RLDR TRANSP MTP FORT.LB<br>TRANSP<br>DELETE/V TRANSP.SV<br>RLDR DMULT FORT.LB<br>DMULT<br>DELETE/V DMULT.SV FT.TX | | | AIT Back matrix macro for 1860 x 64 matrix. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| TRACKING PROGRAMS | AITGOF | SCS(SCSGOF), POWER | | This program is not likely to be needed again. It was used to test the consistancy of the gain-offset (linear) generalization of the SCS algorithm. This variant of AITSCS calculates the gains and offsets derived from 16 different pairs of linear equations for each of x and y, each with one of the pair assuming a zero x and y displacement and the other of the pair using a different displacement. A point source is assumed. This is done for each tracker cycle. |
| | AITSCS | SCS, POWER | | Program to calculate gain factor, offset term, and tracker cycle mask for elliptical nutation pattern derived for AIT. This program calculates gain and offset parameters for SCS-like tracking using elliptical nutation pattern. The program also calculates usable arcs for given object sizes, and axis crossing points for objects of given displacements. This information is written to files BEGEND and GAINOFF. This program outputs to the terminal, and has all the hook to enable output to file PRINTER. This program assumes point source. It is not as accurate in defining gain and offset as program GOCALC. |
| | AITTRK | SCSTRK, POWTRK | | Program to calculate tracker displacements from a "detector vector" waveform file. The file name is determined by user prompt. The gains and offsets and beginnings and ends of the arcs are obtained from files GAINOFF and BEGEND respectively. |
| | BEGLST | NONE | | This program reads data file BEGEND and writes the information recovered from this file out to file PRINTER. BEGEND contains the beginning point, the two axis crossing points, and the end point for each arc of each cycle. The file BEGEND may have been generated by any of a number of programs. |
| | GNLST | NONE | | This program reads file GAINOFF and prints the formatted contents out to file PRINTER. GAINOFF contains the gains and offsets for each tracker cycle, as obtained from some program that generates them (see AITSCS or GOCALC). |
| | GOCALC | GOSCS, GOPOW | | Program to calculate gain factor, offset term, and tracker cycle mask for elliptical nutation pattern derived for AIT. This information is stored in data filed, BEGEND and GAINOFF. This program uses a different, more natural (and successful) method of deriving these parameters than that employed by program AITSCS. The method is also simpler to code. The program uses detector vector waveforms (simulated or real) for which the centroid of the intensity distribution are known. Using two of these by applying the AIT tracker algorithm to them supplies the two linear equations required to find the gain and offset. By playing with parameters such as object size and shape and determining what effect these factors have on the gain and offsets calculated for the different trackers cycles, information about the weaknesses and strengths of the generalized SCS algorithm can be gained. |
| | PIXGRPH | GEM5GRAPH, PGPDR, FIND | | This program prompts the user for an input file, preferably one created by TRKRUN. It then presents the information contained in that file graphically. This information is a set of AIT-tracker transfer curves, one for x displacements and one for y displacements. (See TRKRUN documentation). |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| TRACKING PROGRAMS | PIXTRNS | SCSTRK, POWTRK | | This is a special purpose version of TRKRUN which selects the diagonal pixels and creates a file called PIXTRK that you can feed into program PIXGRPH to graphically display tracker cycle transfer curve. For further documentation, see program TRKRUN |
| | TIMELINE | NONE (uses file BEGEND created by AITSCS or GOCALC) | | Program prints out a graph on the terminal and to file PRINTER detailing the beginning and endpoints of tracker arcs for the AIT SCS tracker algorithm. |
| | TRKGRPH.FR | PGPDR, FIND, GEN6GRAPH | | This program prompts the user for an input file, preferable one created by TRKTRNS. It then presents the information contained in that file graphically. This information is a set of AIT-tracker transfer curves, one for x displacements and one for y displacements, for each relevant tracker cycle. (See TRKRUN documentation). |
| | TRKRUN.FR | SCSTRK, FOWTRK | | This program produces tracker transfer functions, using as inputs the pixels of the forward matrix (formed from detector vector basis set). Each run scans one row (for x displacements) and one column (for y displacements). This program allows the user to choose which row and column of the 64 pixel field will be chosen. Transfer functions are output to file PIXTR. |
| | TRKTRNS.FR | SCSTRK, POWTRK | | This program uses as input a set of 16 waveforms. The files containing the waveforms all have a similar name (they are created by program SIMRUN): their first four characters of the names are the same for all the files, but, the fifth and sixth characters are two digits between "01" and "16". The waveforms in the files are input, one at a time, and in order determined by the 5th and 6th characters, and the x and y tracker displacements are calculated and output to a file named TRKR. TRKR may then be used as input to TRKGRPH or it may be used as input to TRNSGRPH. For further documentation on how tracker algorithm works, see program AITTRK. |
| | TRNSGRPH | GEN6GRAPH, PGPDR, FIND, PGP, WAITLOOP, PLAYBK | | This program prompts the user for an input file, preferably one created by TRKTRNS. It then presents the information contained in that file graphically. This information is a set of AIT-tracker transfer curves, one for x displacements and one for y displacements. (See TRKTRNS documentation). It is assumed that the name of the TRKTRNS created file is a 4-character name. It is also assumed that another input file exists, with a name which is identical to the other input file in its first four characters, and has MV as its fifth and sixth characters. This ----MV file is assumed to have been created by CIMAGE as a movie with 16 frames. These frames portray the positions of the object represented by the 16 waveforms as it is moved through the tracking positions (see program SIMRUN). These are displayed on the Genisco monitor by subroutine PLAYBK. This movie precedes the display of the AIT tracker transfer curves. Preceding the display of the movie, another picture is shown. This image is contained in a file ----I This file contains the intensities of the 64 pixels, and is also created by program CIMAGE. This picture is displayed by subroutine PGP. |
| | | | | |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| GENISCO PROGRAMS | BFTS (Assembly language) | | | This assembly language "subroutine" is used as a table in the Genisco diagnostic program STPD. The save file for STPD is renamed DIAG.SV. |
| | BITS (Assembly language) | | | This assembly language "subroutine" is used as a table in the Genisco diagnostic program STPD. The save file for STPD is renamed DIAG.SV. |
| | CONVERT | NONE | | Prompts user for name of file. The program was originally designed to convert 256 word binary files of Genisco diagnostic codes to ASCII files containing the corresponding ASCII octal representations of these 256 numbers with line feeds between each pair of numbers. The program can easily be converted to a more general purpose routine. This program created octal numbers in programs BFTS, BITS, INTS, WLTS, WTDS. Warning: There is another program with the name CONVERT in other directories. This other program is used in generating an AIT back matrix. |
| | DIAGUNPACK | NONE | | This is a program to unpack the Genisco data file, DIAGNOSTIC, supply a missing word at the end of each 256 word block, and write these blocks out to appropriately named disk files. The program CONVERT is then to be used to convert the numbers in these files to an ASCII representation that can be used to create input files for the assembler. The assembled files are used in an assembly language program, DIAG.SV. (See program STPD) |
| | FIXFRAME | PGPDR, FIND | | Subroutine to display images on Genisco color system monitor rapidly in a slide projector like fashion. The frames may be displayed in any order the user chooses. These images have been previously stored in a data file (the name of which is requested by a prompt in the program). The format of the data file is: 26 integers per image frame. The 26 words correspond to the data required by the Genisco PGP to display an image. |
| | GNEDIT | NONE | | This program replaces the missing word at the end of each 256 word block of the original Genisco operating system binary file. It also allows one to use two different versions of this file, the two differing in only the very first three words of the file. This is because, on the tape, these three words were zeroes, but in the listing that came from Genisco, they were not. This program is hopefully obsolete, since any program named GENOP which has 10000 (octal) bytes has already been processed by this program. But if, through some terrible catastrophy, it is needed, here it is. |
| | GNLDE | NONE | | This version is to be used in the non-memory-mapped environment. This very special, stand alone version of PGPDR is used to boot the PGP. It transfers the contents of a 10000 (octal) word file, G, to the PGP. This file is the Genisco operating system, on file GENOP, renamed to file G. GNLDE must be loaded into the PGP before any other Genisco routines can be called. A successful load of the operating system is indicated by the appearance of three rectangles in the upper left corner of the color monitor. Another version of this program, GNLDM, is for a memory mapped environment. |
| | GNLDM (assembly language) | NONE | | This version is to be used in the memory-mapped environment. This very special, stand alone version of PGPDR is used to boot the PGP. It transfers the contents of a 10000 (octal) word file, G, to the PGP. This file is the Genisco operating system. GENOP is ren[illegible] G. It must be run before any other Genisco routines can be called. A successful load of the operating system is indicated by the appearance of three rectangles in the upper left corner of the color monitor. Another version of this program, GNLDE, is for a non-memory mapped environment. |
| | | | | |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| GENISCO PROGRAMS (cont.) | INTS (Assembly language) | | | This assembly language "subroutine" is used as a table in the Genisco diagnostic program STPD. The save file for STPD is renamed DIAG.SV. |
| | NUTPAT | WAITGRAPH, PGPDR, FIND | | Program to display the elliptical nutation pattern on the Genisco color system. There is a mechanism to allow slow motion generation of the display using a wait-loop parameter. |
| | PAL | FIND, PGPDR | | Program to generate, on color monitor, either of two displays: 1) the hardware generated color pallet described in the Genisco programming manual; 2) the 256-word color video look up table that is contained in the video color memory at the time. |
| | PGP3.FR | PGPDR, FIND | | Program to display image file on the Genisco color system. This program prompts user to supply the name of the file containing the image (a 64 integer file containing a pixel vector). The image is displayed in a pseudo-color representation, with fifteen intensity levels. A key, defining the color-intensity mapping, is displayed under the 8 x 8 pixel image. Labeling is included. Note: PGP scales the vector so that at least one maximum intensity pixel is in the display. |
| | PGPT.FR | FIND, PGPDR | | Uses Detector vector file S0918$2334. Program to test the frame rate of a Fortran program designed to send the PGP images, that is, 64 pixel values to be displayed as a level sliced, pseudo-color coded, 8 x 8 pixel field. PGPT.FR creates 64 frames, each frame displaying one dark pixel. The frames are created in an order that gives an appearance of the dark pixel moving through the pixel field. |
| | PLAYBACK | PGPDR, FIND | IMAGE | Subroutine to display images on Genisco color system monitor rapidly in motion picture like fashion. These images have been previously stored in a data file (the name of which is requested by a prompt in the program). The format of this file data is: 26 integers per image frame. The 26 words correspond to the data required by the PGP to produce an image. |
| STPD.SR | STPD (Assembly language) | | | This is the main program in the Genisco diagnostic package. It uses INTS, BFTS, BITS, VLTS, as data to send to the PGP. These data packages are the diagnostic modules, in PGP assembly language. STPD ships these modules out to the PGP and reads back the results. For more detail on the operation of this program, see the main (orange looseleaf notebook) Genisco doucmentation manual. The save file STPD.SV is renamed DIAG.SV. Note: File STPD is the load module for this program. |
| | SPIRPAT | WAITGRAPH, WAITLOOP, PGPDR, FIND | | Program to display the linear spiral nutation pattern on the Genisco color system. There is a mechanism to allow slow motion generation of the display using a wait-loop parameter. |
| | VLTS (Assembly language) | | | This assembly language "subroutine" is used as a table in the Genisco diagnostic program STPD. The save file for STPD is renamed DIAG.SV. |
| | WOTS (Assembly language) | | | NOTE: At present, we do not have the hardware associated with this test; therefore it is not used. This assembly language "subroutine" is used as a table in the Genisco diagnostic program STPD. The save file for STPD is renamed DIAG.SV. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| BENCH | AITLD.FR | DTA4F, SENDWORD, TINA, TOUTA | | Program to load AIT rack with elliptical nutation pattern. Frequency and gains are specified by the user. An operationally similar program for the Apple, written in Basic +6502 assembly language exists on mini-floppies, to be run on 48K Apple computer with brand name "Mountain" D-A /A-D board. |
| | EXPDSK.FR | ATODFW1, HEADGN, PHEAD, ILIPSS, INCIG, INCIR, INSQ, INDSK, NCON, ROUTINES IN DSCUTL1.LB, WRHEAD, WRDATA, NAMFIL, CFIL, ERRCK | | Program to collect optical breadboard data for the AIT program. Data is collected for the specified number of nutation cycles and stored on disk for experimentation by program IMAGE. This program is the complement to SIMDSK, which simulates the quad cell output waveforms and writes them to disk. The header data is generated in the same manner as in SIMDSK. Program to take data from bench and store in file named by NAMFIL routine (date and time determined). Generates standard SIMDSK file format with header. Header may be changed by user with standard SIMDSK prompts. Data obtained from bench (465 time samples, 4 quad outputs per time sample) is stored in standard waveform vector format. WARNING: In subroutine which actually drives DGDAC (subroutine ATODFW1.SY) all interrupts have been disabled. Therefore, if all data synchronization clock signals are not received, the program will hang forever. Therefore the rack must be on and the cables correctly connected before this program is run. Also, this program cannot be run with a memory mapped operating system. Another version of this program, MEXPEDSK has been altered so that it runs the data converter in a memory-mapped environment. |
| | HWLOAD.FR | DA4F, SENDWORD, TINA, TOUTA | | Program to load AIT rack with pure sin and cos waves. Uses 32 cycles per rack image, allowing 64 points of resolution per cycle. User prompts allow selection of frequency and gain of trig waves. Same Apple note as AITLD.FR. |
| | MSNAP.FR | ATODFW1, FSHEADGN, NCON, PEHAD. Also requires the availability of the memory map compatible version of FSIMAGE for program swap. | | CREATES DATA FILE SNAPDATA. SNAP is the first program in a Fortran program swap. It works precisely in the same manner as EXPDSK, except that its output data file is named SNAPDATA., and it passes control to a version of IMAGE., named FSIMAGE. FSIMAGE processes the waveform vector into an image vector and displays this image on the Genisco. After permitting use of the standard image utilities, control of program environment is returned (swapped back to) SNAP, to allow more data to be taken. This "infinite loop" may be escaped by using a "control- A" program interrupt. WARNING: In subroutine which actually drives DGDAC (subroutine ATODFW1.SR) all interrupts have been disabled. Therefore, if all data synchronization clock signals are not received, the program will hang forever. Therefore the rack must be on and the cables correctly connected before this program is run. Also, this program can only be run with a memory mapped operating system. |
| | SNAP.FR | ATODFW1, FSHEADGN, NCON, PHEAD, WRDATA, WRHEAD. Also requires the availability of FSIMAGE for program swap. | | CREATES DATA FILE SNAPDATA. Program to collect optical breadboard data for the AIT program. Data is collected for the specified number of nutation cycles and stored on disk for experimentation by IMAGE. This program is the complement to SIMDSK, which simulates the quad cell output waveforms and writes them to disk. The header data is generated in the same manner. SNAP is the first program in a Fortran program swap. It works precisely in the same manner as EXPDSK, except that its output data file is named SNAPDATA, and it passes control to a version of IMAGE, named FSIMAGE. FSIMAGE processes the waveform vector into an image vector and displays this image on the Genisco. After permitting use of the standard image utilities, control of program environment is returned (swapped back to) SNAP to allow more data to be taken. This "infinite" loop may be escaped by using a "control-A" interrupt. WARNING: In subroutine which actually drives DGDAC (subroutine ATODFW1.SR), all interrupts have been disabled. Therefore, if all data synchronization clock signals are not received, the program will hang forever. Therefore the rack must be on and the cables correctly connected before this program is run. Also, this program cannot be run with a memory mapped operating system. Another version of this program MSNAP, has been written to run in the memory-mapped environment. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| <u>BENCH</u> (cont) | STRT.FR | DIA4F, TINA, TOUTA | | Program to change frequency, gain, or start or stop AIT rack. May be used after rack is loaded with nutation pattern by AITLD. (If rack is loaded by HWLOAD, STRT must be changed so that the parameter "M" in both STRT and HWLOAD have the same value.) First prompt asks frequency. Then one is asked if change of gain is required. After setting gain (or not) one is prompted to strike any key to start rack. Then one may strike any key to stop rack. Then one is asked if gain change is desired, etc. Same Apple note as AITLD. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| PMP Support Programs | ASSDIAG | PMPDIAG PMPIO WRHEX | | This program is a version of program PMPDIAG, created in an attempt to solve a communication problem between the Nova and the PMP. The operating system was not passing values which could be interpreted as the ASCII control characters 'CONTROL-Q' and 'CONTROL-S'. It would intercept and throw away these characters when reading them with a FORTRAN 'Read Binary', and then the program would hang, waiting for the PMP to send another word only on a one byte prompt from the Nova. Therefore, it would also hang. This program attempted to circumvent the problem by reading the bytes from an Assembly language routine PMPIO. This did not solve the problem, since the RDOS test for 'CONTROL-S' and 'CONTROL-P was taking place at a "lower" level. (See PMPIO). This program lets the PMP do all the work. It just prints out error messages that the PMP sends to the Nova. |
| | BOOTLD | NONE | | Program to take PMP bootstrap code in file BOOTCODE, and send it to $PMP RS232 Port. Then the microcode program in file PROGCODE is shipped to the PMP. If the front panel lights on the PMP indicate that the load was successful, the user may then load memory with user determined file. The memory to be loaded determined by the user through a program prompt. The user is then presented with further prompts, permitting further PMP downloads. This program may be used to load boot and microcode program, using 'CONTROL-A' to leave this program after a successful load. |
| | BOTHPGPLD.FR | PMPSHIP CKSM | | This fairly complicated program is a vehicle to download to the PMP all the data (except the boot and microcode program, which are loaded by program BOOTLD) required by the PMP to run the optical bench, do real-time tracking, and do imaging for the linear galvanometer AIT system. Some of the information is picked up from files and shipped directly to the PMP. Some of the data is generated totally by this program, and some is picked up by this program and manipulated into the form required by the PMP microcode. Most of these manipulations will make sense only if you are fairly familiar with the system architecture of the PMP and also are familiar with the AIT project and the AIT microcode. Subroutine PMPSHIP ships the contents of the arrays that are passed to it to the PMP with the properly formatted commands and a CHECKSUM, calculated by subroutine CKSM. The start command for these operations and the program to receive the imaging information from the PMP and display this information on the Genisco monitor is the program ?GPWRT. |
| | BTSTRP | MKBYTES,MHEX, MBITS | | Program to read object file generated by microassembler and download microwords, correctly formatted, to PMP via RS232 port. It is assumed that the microcode contains exactly 16 microwords, each 96-bits wide, formatted by the meta-assembler as modified by Bruce Cutler in Winter 1981-1982. BOOTCODE is the output file with the binary code; BOOTSTRAP.OB is the input file with the ASCII object code. |
| | CMATTST | MEMHEX,MKBYTES | | |
| | CMEMRUN | WRHEX,CKSM | | This program is a diagnostic tool used to test so-called C-MEMORY boards in the PMP. The column lengths of this memory is stored in variable NWORDS. Packets containing NWORDS of data for the memory are shipped to the PMP and sent to the column determined by the address in variable IISCOM. A menu of possible data patterns is presented to the user. The user also chooses the column address (as an integer between 1 and 63). The data shipped to memory is then read back into this program and the two patterns are compared. Any discrepancies are described to the user. Further prompts allow repetition of this scheme. Hexadecimal inputs and outputs are used in appropriate places. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| PMP Support Programs | | | | |
| | [illegible]VERTED | PMPSHIP, CKSM, ULMRD | | This program is an early version of PGPLD, which in turn is an early version of BOTHPGPLD. These programs are useful in that they constitute progressively more demanding tests of the PMP when they are run with the appropriate microcode routines in the PMP. For further documentation, see program BOTHPGPLD. |
| | DARPRINT | NONE | | This program is used to produce printouts of various data patterns that are shipped to the PMP. [illegible] program was used for debugging purposes, and may be further hacked up and so used. Output is to file PRINTER. |
| | DBASMAP | CKSM | | NOTE: This program is essentially obsolete, except as it can quickly create a file with 1024 data words in a floating one pattern. It was initally used to create this pattern to test the DAR map of the PMP.<br>This program downloads to the PMP 64 sets of 16 words, each word with one bit set, all others cleared. It requests PMP 'ADDRESS - 1' from user. It uses the (at the time) proper format for the microcode controlled load of PMP memories. IOUT is the array that will be shipped to the output file, DBAMAP. |
| | DECHEX | WRHEX | | This program is a trivially simple but occasionally useful decimal-to-hex converter. |
| | DOUGDIAG | NONE | | A test version of program PMPDIAG which calls COMERR to attempt to get a handle on PMP-Nova communication problem. See programs ASSDIAG and PMPDIAG and COMERR. This program lets the PMP do all the work. It just prints out error messages that the PMP sends to the Nova. |
| | FMATTRK | SCSIMAT,POWFMAT | | Version of program AITTRK which produces a pretty printout used in debugging the PMP implementation of the AII SCS tracker algorithm. In its present form, this program uses the foward matrix, FMAT as its input file; that is, its source of detector vectors. |
| | MAP [illegible] | NONE | | Program to take PMP BOOTSTRAP code in file BOOTCODE, and send it to SPMP RS232 port. This is a version of BOOTLD that does not prompt user to provide a memory file download, but instead chains to program PMPDIAG.SV. This program may be used to load boot and microcode program using 'CONTROL-A' to leave this program after a successful load. |
| | MATMULT | WRHEX,FMOP, ULMRD | | This is an earlier version of PGPMULT.FR. This version differs from PGPMULT.FR in that it does not use the Genisco programmable graphics processor display. There was some kind of DG bug in the way the relocatable reloader was operating which caused FMOP to return nonsense, an effect dependent on its location in the reloader line. Because of this problem, the graphic display of the PGP was chosen as a more convenient tool. This program is kept around as a possible future AII tool, and also, as a possible tool to be used to examine the DG bug. The program flow is not too complicated, but for further documentary reference, see PGPMULT.FR. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| PMP Support Programs | MATMULTO | WRHEX,CKSM, FMDP,COMERR, ULMRD | | NOTE: This was a version of program MATMULT created to analyze the cause of the problem later diagnosed as a DG bug. Then this program was changed (FMDP call was commented out) so that it could be used as a debugging routine on the PMP. For further documentation, see PGPMULT.FR.<br>This is an earlier version of PGPMULT.FR. This version differs from PGPMULT.FR in that it does not use the Genisco Programmable Graphics Processor display. There was some kind of DG bug in the way the relocatable reloader was operating which caused FMDP to return nonsense, an effect dependent on its location in the reloader line. Because of this problem, the graphic display of the PGP was chosen as a more convenient tool. This program is kept around as a possible future AIT tool, and also as a possible tool to be used to examine the DG bug. The program flow is not too complicated, but for further documentary reference, see PGPMULT.FR. |
| | MEMTST | MKBYTES,MEMHEX, WRHEX,ULMRD | | NOTE: This program is a variant of program CMATTST. It is an adaptation of that program to test other memories, in this case, the main memory. The command code in variable ICOM tells the PMP which memory to test and which data path to take in the test. This version uses ICOM=6 which tests the main memory using the Y-BUS. (These conventions are determined in the microcode.)<br>This program is a diagnostic tool used to test so-called 'C-MEMORY' boards in the PMP. The column lengths of this memory is stored in variable NWORDS packets containing NWORDS of data for the memory are shipped to the PMP and sent to the column determined by the address in variable IISCOM. A menu of possible data patterns is presented to the user. The user also chooses the column address (as an integer between 0 and 63). The data shipped to memory is then read back into this program and the two patterns are compared. Any discrepancies are described to the user. Further prompts allow repetition of this scheme. Hexadecimal inputs and outputs are used in appropriate places. |
| | MEMTSTBB | MKBYTES,MEMHEX, WRHEX,ULMRD | | NOTE: This program is a variant of program CMATTST. It is an adaptation of that program to test other memories, in this case, the main memory. The command code in variable ICOM tells the PMP which memory to test and which data path to take in the test. This version uses ICOM=11 which tests the main memory using the B-BUS. (These conventions are determined in the microcode.)<br>This program is a diagnostic tool used to test so-called 'C-MEMORY' boards in the PMP. The column lengths of this memory is stored in variable NWORDS. Packets containing NWORDS of data for the memory are shipped to the PMP and sent to the column determined by the address in variable IISCOM. A menu of possible data patterns is presented to the user. The user also chooses the column address (as an integer between 0 and 63). The data shipped to memory is then read back into this program and the two patterns are compared. Any discrepancies are described to the user. Further prompts allow repetition of this scheme. Hexadecimal inputs and outputs are used in appropriate places. |
| | MICRO | MKBYTES,MHEX, MBITS | | Program to read object file generated by microassember and download microwords, correctly formatted, to PMP via RS232 port. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| PMP Support Programs | NEWCMAT | MKBYTES, MEMHEX, WRHEX | | NOTE: This program is an enhancement of program CMATIST. NEWCMAT has the test run through all the columns, therefore the prompt requesting the column number has been commented out. <br> WARNING: The subroutine for reading the information sent back from the PMP, ULMRD, has not been installed in this program, so the 'CONTROL-Q', 'CONTROL-S' RDOS operating system bug is still in this program, limiting its usefulness. ULMRD was never installed in this program (it replaces binary reads and writes to the PMP) because the C-Memory was proven reliable by other means (doing matrix multiplies). <br> This program is a diagnostic tool used to test so-called 'C-MEMORY' boards in the PMP. The column lengths of this memory is stored in variable NWORDS. Packets containing NWORDS of data for the memory are shipped to the PMP and sent to the column determined by the address in variable IISCOM. A menu of possible data patterns is presented to the user. The user also chooses the column address (as an integer between 0 and 63). The data shipped to memory is then read back into this program and the two patterns are compared. Any discrepancies are described to the user. Further prompts allow repetition of this scheme. Hexadecimal inputs and outputs are used in appropriate places. |
| | PGPLD | PMPSHIP CKSM | | This version of PGPLD is virtually identical to BOTHPGPLD (in fact, I cannot see any difference between them, though there may be one). This fairly complicated program is a vehicle to download to the PMP all the data (except the boot and microcode program, which are loaded by program BOOTLD) required by the PMP to run the optical bench, do real-time tracking, and do imaging for the linear galvanometer AIT system. Some of the information is picked up from files and shipped directly to the PMP. Some of the data is generated totally by this program, and some is picked up by this program and manipulated into the form required by the PMP microcode. Most of these manipulations will make sense only if you are fairly familiar with the system architecture of the PMP and also are familiar with the AIT project and the AIT microcode. Subroutine PMPSHIP ships the contents of the arrays that are passed to it to the PMP with the properly formatted commands and a checksum, calculated by subroutine CKSM. The start command for these operations and the program to receive the imaging information from the PMP and display this information on the Genisco monitor is the program PGPHRT. |
| | PGPMULT | PGP, PGPOR, FIND, CKSM, ULMRD | | This program sends a detector vector to the PMP, optionally loads the C memory with the back matrix (under user control), sends the proper run command to the PMP, reads back the image vector (pixel vector) from the PMP, and displays the image on the Genisco color system. The detector vector is obtained from a detector vector file, the name of which is obtained from the user by a program prompt. |
| | PGPMULT64 | PGP, PGPOR, FIND, CKSM, ULMRD | | NOTE: This is a variant of program PGPMULT. The only difference between this program and PGPMULT is that this program displays the 64 detector vectors obtained from the forward matrix, one after another, rather than a single detector vector obtained from a user determined file, as in PGPMULT. <br> This program sends a detector vector to the PMP, optionally loads the C-memory with the back matrix (under user control), sends the proper run command to the PMP, reads back the image vector (pixel vector) from the PMP, and displays the image on the Genisco color system. The detector vector is obtained from a detector vector file, the name of which is obtained from the user by a program prompt. |
| | PGPRUN | | | NOTE: This program is exactly similar to PGPMULT64, except that it is made to run forever (by looping). For documentation, see program PGPMULT64. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| PMP Support Programs | PGPWRT | PGPIDM PGPFAST FPGPDM FIND PGPTI JLMRD | | This program was written to be used with program PGPLD (or BOTHPGPLD). After PGPLD has downloaded all the information required by the PMP to do its work, PGPWRT sets up the image display hardware (using subroutine PGPTI) and then gives the PMP the 'RUN' command. This program then receives the 64 element pixel vector from the PMP and ships it to the display system (the Genisco color display system). The program then returns to waiting for the next pixel vector to be received from the PMP. This goes on until interrupted by a 'CONTROL-A'. PGPWRT is a dedicated program, with some special purpose subroutines that are variants of some standard, heavily used AIT subroutines. This specialization is needed to solve a timing problem: the PMP microcode, for efficient machine operation, needs to send each pixel vector element to the Nova, as soon as it is calculated (that is, at the end of each matrix column X detector vector dot (inner) product). Therefore, the interval between the receipt of the last element in one pixel vector and the receipt of the first element of the next pixel vector is very short. This is the interval of time permitted for the scaling and data packing of the imaging information, and the transporting of this data package to the PGP. This process had to be streamlined. Further, the Assembly language driver, PGPDM, has a waiting loop in its code which requires the 'Ready to Receive' interrupt from the PGP before the loop is exited. This loop had to be eliminated in the driver FPGPDM. |
| | PMPDIAG | NONE | | This program lets the PMP do all the work. It just prints out error messages that the PMP sends to the Nova. |
| | PMPDIAG.2 | PMPDIAG,PMPIO, URHEX | | This program is a version of program PMPDIAG, created in an attempt to solve a communication problem between the Nova and the PMP. The operating system was not passing values which could be interpreted as the ASCII control characters 'CONTROL-Q' and 'CONTROL-S'. It would intercept and throw away these characters when reading them with a FORTRAN 'Read Binary', and then the program would hang, waiting for the PMP to send another word. The PMP was programmed to send another word only on a one byte prompt from the Nova. Therefore, it would also hang. This program lets the PMP do all the work. It just prints out error messages that the PMP sends to the Nova. |
| | SIMMEM | CKSM | | This program creates files to be sent to the PMP. These files load the nutation pattern for sending drive voltages to the Galos. Possible user defined phase shift included. The name of the output file is IMMEM. It is picked up and downloaded to the PMP by several programs. |
| | STMEMF0 | CKSM | | This program downloads to the PMP 64 sets of 16 words, each word with one bit cleared, all others set. It requests PMP 'ADDRESS - 1' from user. It uses the (at the time) proper format for the microcode controlled load of PMP memories. IOUT is the array that will be shipped to the output file, THEMF0 the file this program creates may be used with program BOOTLOAD to test PMP memory. (See the end of program BOOTLD.) |
| | STMEMF1 | cksm | | This program downloads to the PMP 64 sets of 16 words, each word with one bit set, all others cleared. It requests PMP 'ADDRESS - 1' from user. It uses the (at the time) proper format for the microcode controlled load of PMP memories. IOUT is the array that will be shipped to the output file, THEMF1 the file this program creates may be used with program BOOTLOAD to test PMP memory. (See the end of program BOOTLD.) |
| | | | | |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| PMP Support Programs | ULMTST.FR | NONE | | This version of PMPDIAG is used to test the ULM board using Assembly Language subroutine ULMRD. It examines and displays the status words on error. This program lets the PMP do all the work. It just prints out error messages that the PMP sends to the Nova. |
| | VECTLD | PMPSHIP,CKSM, ULMRD | | This program is an early version of CVECTLD, which is an early version of program PGPLD, which in turn is an early version of BOTHPGPLD. These programs are useful in that they constitute progressively more demanding tests of the PMP when they are run with the appropriate microcode routines in the PMP. This version asks the PMP to find only one displacement (the X displacement) of only one tracker cycle (tracker cycle 10). The user is prompted for the name of the detector vector file to be so analyzed. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | OVERLAY OCUPL ACCPL | NONE | IMAGE | This subroutine finds the average energy in the waveform of each quadrant and subtracts this average from each value of the corresponding quadrant's waveform. |
| | OVERLAY OACFIL ACFILT | NONE | IMAGE | When the parameters are correctly set, this subroutine removes the DC offset from a waveform vector on a quadrant-by-quadrant basis. For each quadrant, the offset is found by looking at the energy in that quadrant at points in the nutation pattern when the pattern is farthest from that quadrant. Theoretically, the average of these points can represent the DC bias. The algorithm below is a straightforward application of this theory. |
| | ACFLGN | NONE | ACGEN | This routine is called by ACGEN once per nutation point. It tests the point to find if it is a current extreme point from any quadrant, and if it is, it has this point replace the appropriate obsolete extreme point and moves the points in storage arrays, RPTS and NPTS, so that the values are always stored with the least extreme values first. It does this on a quadrant-by-quadrant basis. |
| | ALG2 | MMULT | IMAGE | This routine receives from the calling program, a buffer containing detector data (either real or simulated). It then treats this buffer as a vector and multiplies this vector with the appropriate back matrix transpose. The product is returned to the calling program with the detector data left unchanged. This product is a "pixel vector" representing the pixel intensities in an "image" reconstruction. This routine calls a matrix multiplication subroutine, MMULT, which does the actual multiplication. See the header on that routine for explanation of how it works. |
| | ALG2FM | FMDP | IMAGE | This routine receives from the calling program, a buffer containing detector data (either real or simulated). It then treats this buffer as a vector and multiplies this vector with the appropriate back matrix transpose. The product is returned to the calling program with the detector data left unchanged. This product is a pixel vector for an image. This routine calls a fast matrix multiplication subroutine, FMDP, which does the actual multiplication. FMDP is an assembly language routine (see FMDP.SR for calling sequence), which does its own disk I/O. It cuts processing time from about 5 minutes to about 6 sec. |
| | ATODFW1.SR | NONE | | Routine to initiate A/D conversions with DGDAC but sans SAM. DMA mode of data collection is used. This version is for a non-memory mapped environment. The A to D conversions are started on a signal from the rack and each sample is clocked on signals from the rack. DGDAC ≡ Data General Analog to Digital Converters. SAM ≡ Data General Sensory Access Monitor Software Package. It is very slow. This routine is FORTRAN callable. The calling parameters are as follows: NPT - Number of Conversions<br>N256 - Number of Blocks of 256 PTS to Convert<br>NREM - Number of Remaining Conversions After 256 Blocks Done<br>NCHAN - Number of Channels to Convert<br>ICHAN - Beginning or Only MUX Channel Number.<br>CLK - Clocking Mode Selector (EXT or INT)<br>0 = EXT, 1 = INT for Single Channel<br>DATA ADDRESS - Address at Which Data is to be Stored |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | ATODFWM.SR | NONE | | Routine to initiate A/D conversions with DGDAC but sans SAM. SAM ≡ Data General Sensory Access Monitor Software Package. It is very slow. DMA ≡ Direct Memory Access. DMA mode of data collection is used. This version is for a memory mapped environment. Routine is FORTRAN callable. The calling parameters are as follows:<br>NPT - Number of Conversions<br>N256 - Number of Blocks of 256 PTS to Convert<br>NREM - Number of Remaining Conversions after 256 Blocks Done<br>NCHAN - Number of Channels to Convert<br>ICHAN - Beginning or Only MUX Channel Number<br>CLK - Clocking Mode Selector (EXT or INT)<br>0 = EXT, 1 = INT for Single Channel<br>DATA ADDRESS - Address at Which Data is to be Stored |
| | CELL2.FR | NUTLPS OVLAP IBOX INRI | QCELL | This program is called NSTEP/NOUT times per nutation point in the image cycle. These NSTEP NOUT values of Q(1-4) produced in this program are integrated by subroutine INTEG to produce four points in the final detector vector, one for each quadrant (waveform). Arguments are identical to QCELL. All of the work is done here. |
| | CFIL(IER) | ERRCK | NAMFIL | This routine creates a file on the current partition. It is named NAME. If the file already exists, the low order ASCII digit of the name is incremented. When it reaches 9, the next highest digit is incremented. |
| | CKSM | NONE | Almost all PMP Download-Data Creation Routines | This program is a FORTRAN callable routine to calculate the CHECKSUM required by PMP data packets. There are two parameters passed to this subroutine, IWORD and ICLEAR. IWORD contains the next integer value to be added to the CHECKSUM. If ICLEAR is not equal to zero, the CHECKSUM, stored in this program, is set to zero. If ICLEAR is equal to zero, then the current value of the CHECKSUM is stored in this routine and also returned to the calling program in the variable, IWORD. Obviously, this subroutine is not re-entrant. |
| | CMPR | GENSGRAPH, PGPOR, FIND | IMAGE | Routine to compare the present detector vector with another detector vector on file in the user's directory. The comparison is made graphically on the Genisco monitor. There are two modes of display. If a particular quadrant is requested, three graphs will be presented. The first is the newly requested detector vector (in ARRAY ICOMP), the second is the old detector vector (in IBUF), and the third is the difference of the two. If the user answers "5" to the prompt, there will be 4 graphs presented to the screen. They will be the difference graphs for each of the four quadrants. A carriage return answer to the request for the name of a file will preserve the present ICOMP buffer. |
| | CMPR2 OVERLAY OCMPR | GENSGRAPH, PGPDR, FIND | FSIMAGE | This is a variant of CMPR. See that program for documentation. This differs from CMPR in that no difference graph is shown. In Option 5, the ICOMP detector vectors are graphed. |
| | COMERR | NONE | | This program was written as a diagnostic tool to help understand a communication problem we were ha[illegible] with the RDOS operating system. The operating system was not passing values which [illegible] be interpreted as the ASCII control characters CONTROL-Q and CONTROL-S. [illegible] intercept and throw away these characters when reading them with a FORTRAN 'READ [illegible]' and then the program would hang, waiting for the PMP to send another word. The PMP was [illegible]rammed to send another word only on a one byte prompt from the NOVA. Therefore, it would [illegible] hang. This program attempted to read the status of the ULM at the time of failure. [illegible] were also being afflicted with 'READ BINARY' erro[illegible] which would send control of the program to a location where this subroutine was called. This problem was later solved. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | COND | NONE | FSIMAGE, CIMAGE | Routine to condition the waveforms in various ways. |
| | CTLC | none | A number of programs | This subroutine enables the user to interrupt the program with a CONTROL-C and to later use the BREAK.SV file created by the CONTROL-C to restart the program at the point of interruption. |
| | DECOASC | NONE | | Subroutine to convert from binary to ASCII values. Input if a 16 bit quantity found in IVAL Output is into the 3 word quantity, JVAL. A sign and 5 digits are returned. |
| | DTA4F | NONE | | Data General data access and control digital to analog. This routine sends output to DG/DAC D/A converter, without SAM (Sensor Access Manager Software Package). This routine is FORTRAN IV callable. The routine expects input in 2 calling parameters:<br>ARG1 - Mask for Channel Select. Using 4 least significant bits; a set bit selects channel.<br>ARG2 - 1 - 4 word block containing data in 12 most significant bits. |
| | ERRCK | NONE | | Subroutine to check FORTRAN error return flag. If not one, the flag is converted to the appropriate format for DG/DAC error messages. 3 is subtracted and the number is printed in octal. |
| | FIND | NONE | | This program is passed the name of an array. It returns in variable ISTART, the pointer to the memory location of the first element of the array NAME. (That is, ISTART contains the "logical" address of the first element of array NAME.) |
| | FMDP | NONE | | This program multiplies a vector by a matrix. In its present configuration, the column size is 1860, and the output vector has length 64. The program works with integers, and uses double precision integer multiplies with a double precision integer buffer for the partial sums of each dot product calculation. This program should be passed 3 buffers, one large enough to contain a column of the matrix, one the size of the output vector, and one containing the vector (the same size as the first buffer). The dimensions used by the program are set in the constants section. This program opens matrix file BMAT and handles all the I/O itself, opening a free channel, then closing it again when it is done. |
| | FORW | NONE | | Used by program GFMX in creation of the forward matrix. |
| | FPGPDM | NONE | | This is a version of the FORTRAN callable ASSEMBLY language driver for communicating with the PGP in a memory-mapped RDOS environment. It is for high-speed applications. This version does not wait in loop for interrupt handler to report; therefore, it is to be used when intermediate processing is necessary and this processing will take enough time to prevent another call to Genisco before Genisco is ready.<br>Subroutine PGPDR is an interface between a Nova RDOS program, written in FORTRAN, and the Genisco GCT3 operating system. It talks to the Genisco programmable graphics processor. It includes the following user callable routines:<br>DFPGP - Defines the PGP to RDOS<br>RMPGP - Removes the PGP from RDOS<br>WRPGP - Starts a write of graphics file to PGP from Nova<br>WCPGP - Continues write of graphics file to PGP from Nova<br>RDPGP - Reads data from PGP into Nova memory |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | GETDAT | RDHEAD, RDDATA | | These subroutines ar in DSCUTL1.LB. This program is a variant of subroutine GETDAT. It differs from GETDAT in the value of the variable NAMDIR, which is the name of the directory the image-type program is required to work in. Also, the call to NAMFIL is suppressed. The program is required to read data from file SNAPDATA, the output of the bench input program, SNAP. This subroutine picks up header and detector vector information for use by the FSIMAGE program. See FSIMAGE, SNAP. |
| | FSHEADGN.FR | NONE | | Subroutine to generate a header array for a detector vector file. This is a variant of HEADGN.FR that eliminates most of the user prompts. It is used in program SNAP which runs in conjunction with FSIMAGE. FSHEADGN does not issue calls to the usual subroutines. A further trimming could eliminate the one remaining user prompt. |
| | GEN2GRAPH | PGPDR,FIND | | This version is exactly the same as GENSGRAPH, but has a different name so that it can be used in the same overlay file with GENSGRAPH, but in a different overlay. |
| | GENGRAPH Program GEN6GRAPH | PGPDR,FIND | | Subroutine to graph 6 X-Y plots to the Genisco, each in a different color. This version of GENGRAPH is tailored for use with the tracking transfer function graphing programs, TRNSGRPH, TRKGRPH, PIXGRPH. The program is no longer self-scaling in Y, requiring YMIN and YMAX to be passed. Also, the Y = 0 line is printed for each call to this routine. The 6 graphs will appear in same region of the screen. Each graph represents the transfer curve for a single tracker cycle. |
| | GENGRAPH Program GENGRAPH | PGPDR,FIND | | Subroutine to graph an X-Y plot to the Genisco. The graphs will appear in same region of the screen. NOTE: Subroutine FIND returns the initial logical memory address of the first element in the passed array. Further NOTE: There are many variants of this program floating around, each of which have some slight wrinkle or added feature. They should each describe how they differ from this one in their documentation. The names of these variants all end in 'GRAPH'. |
| | GENGRAPH Program GENSGRAPH | PGPDR,FIND | | Subroutine to graph an X-Y plot to the Genisco. This version of GENGRAPH will produce up to four single graphs on the screen in four different colors. It divides the screen vertically into four regions, one for each graph. The first graph (IGR=1) will be at the top of the screen. |
| | GETDAT | RDHEAD, RDDATA,ERRCK | IMGSW | These subprograms are in DSCUTL1.LB. This subroutine picks up header and detector vector information for use by IMGSW program. This differs from GETDAT used by IMAGE in that the name of the file to be opened is passed by the file NAMES (created by program SMDSK) rather than prompted from the keyboard. |
| | GETDAT | RDHEAD, RDDATA,ERRCK | IMAGE,CIMAGE | These subprograms are in DSCUTL1.LB. This subroutine picks up header and detector vector information for use by IMAGE program. |
| | GNOIS | RANF | ICUBE IN SIMDSK | This program adds some noise to the waveform being produced by SIMDSK. The parameters of the noise generation are determined by prompts in subroutine HEADGN. |
| | DDPOW.FR | NONE | DOFALC | This program determines the integrated power read from each detector during an arc of an AIT SES tracker cycle. The input is from a bench or simulated detector vector. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | GOSCS | POWER (GOPOW) | GOCALC | This subroutine determines the X and Y tracker ratios for a given tracker cycle, and optionally writes some intermediate values out to file 'PRINTER'. (This option is usually commented out.) The data file buffer (IBUF) is assumed to contain a detector vector, obtained from the Lench or from sumulation. |
| | GPLOWT | GENSGRAPH, PGPOR, FIND | FSIMAGE, CIMAGE | Routine to plot the 4 time waveforms for AIT image reconstruction program. They are plotted on the Genisco color monitor. |
| | GROGEN | | | Oscilloscope display. Subroutine to generate a visual display of image for AIT program. The display is a rectangular grid of size 'MSIZ' by 'NSIZ'. |
| | GSCSF | NONE | SIMDSK,SMDSK | This routine generates a distribution to be used as an image by SIMDSK. The size is of the "Turned On" area is determined by alpha. This is for the cigar-shaped distribution. |
| | GSPSF | NONE | SIMDSK,SMDSK | This routine generates a distribution to be used as an image by SIMDSK. The size is of the "Turned On" area is determined by alpha. This is for the circular spot-shaped distribution. |
| | GSQSF | NONE | SIMDSK,SMDSK | This routine generates a distribution to be used as an image by SIMDSK. The size is of the "Turned On" area is determined by alpha. This is for the square-shaped distribution. W is the parameter governing the size of the square. |
| | HEADGN | PHEAD,ILIPSS, INDSK,WRHEAD, ERRCK | SMDSK | Subroutine to generate a header array for AIT simulation study. The array is configured as an 80 word array and returned to the calling routine. This program is the version of HEADGN used by SMDSK, which is an "Automatic" version of SIMDSK. SMDSK runs with IMGSW, DSTRB, and DSTB in a macro form. HDGN.FR differs from HEADGN.FR in that the prompts to the terminal have been removed. Therefore, you must set the proper parameter values before compiling HDGN and loading SMDSK takes place. |
| | HDGNP.FR | PHEAD, WRHEAD, ERRCK | PRGHD | Subroutine to generate a header array for AIT simulation study. The default values are set by this routine, and the user may change any of them at run time by issuing a 2 letter command followed by the new value. For the case of titles, the entire array must be entered again. The array is configured as an 80 word array and returned to the calling routine. This program is the version of HEADGN used by PRGHD, which is a program that produces a file that contains only a header (without waveform data) in the format used by the GETDAT subroutine of image. |
| | HEADGN | PHEAD, ILIPSS, INCIQ, INSQ, INDSK, ICIR, WRHEAD, ERRCK | SIMDSK | Subroutine to generate a header array for AIT simulation study. The default values are set by this routine, and the user may change any of them at run time by issuing a 2 letter command followed by the new value. For the case of titles, the entire array must be entered again. The array is configured as an 80 word array and returned to the calling routine. |
| | IBOX | NONE | CELL | Performs simple integration on X. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | COMPILER NOSTACK | QCGEN, QVECT, GNOIS which in turn call: QCELL, CELL, OVLAP, OVLAP NUTLPS, etc. | SIMDSK, SMDSK | This routine returns one cycle of waveforms (before integration by INTEG) to SIMDSK. The information is stored in the time array. This version is shortened to take output from 'TIME' array. |
| | IDIS | GRDGEN | | Subroutine to generate image display on oscilloscope. |
| | ILIPSS | NUTLPS | HEADGN in SIMDSK, SMDSK | This is the initialization subroutine that initializes the parameters for NUTLPS. if the user desires, it can also create the file NUTXY, a file containing the elliptical nutation pattern. |
| | ILIPSS | NUTLPS | SMDSK | To generate elliptical nutation pattern. This is the version of ILIPSS called by SMDSK, the "automatic" version of SIMDSK. It differs from ILIPSS.FR in that the user prompts have been removed and the capacity to produce file NUTXY has been deleted. This subroutine initializes the parameters for the elliptical nutation pattern. |
| | INCIG(COMDST) | NONE | HEADGN in SIMDSK, SMDSK | This program creates and returns the proper distribution title. |
| | INCIR(COMDST) Compiler NOSTACK | NONE | HEADGN in SIMDSK, SMDSK | This program creates and returns the proper distribution title. |
| | INDSK(COMDST) Compiler NOSTACK | NONE | HEADGN in SIMDSK, SMDSK | This program creates and returns the proper distribution title. Initialization subroutine to read intensity distributions stored in disk files. |
| | INRI(X) Integer Function | NONE | Various subroutines in SIMDSK, SMDSK | X is converted to nearest integer. |
| | INSQ(COMDST) Compiler NOSTACK | NONE | HEADGN in SIMDSK, SMDSK | This program creates and returns the proper distribution title. |
| | INTEG | NONE | SIMDSK, SMDSK | Subroutine to integrate time waveform output from ICUBE. |
| | INTRD | NONE | SIMDSK, SMDSK | Subroutine to read an optical intensity distribution from disk. A user prompt is used to determine the name of the disk file containing the distribution. The disk I/O is handled. |
| | INTPSW | NONE | SMDSK | Subroutine to read an optical intensity distribution from disk. A user prompt is used to determine the name of the disk file containing the distribution. The disk I/O is handled. This is the version of INTRD adjusted to run with SMDSK. SMDSK is the "automatic" version os SIMDSK that runs with DSTRB, DSTB, and IMGSW without operator intervention. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINE | | | | |
| | IPLOT | NONE | | General purpose TTY plot subroutine. |
| | IPRINT | NONE | IMAGE | Subroutine to print AIT image array. (Note: IM1 is an obsolete 4x4 pixel vector array.) |
| | IPRTSW | NONE | IMGSW | Subroutine to print AIT image array. (Note: IM1 is an obsolete 4x4 pixel vector array.) |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | LSPLOT(KFLG) | | | Used by PLTWV (See PLTWV.FR). This routine plots the data passed to it by the program PLTWV on the XY flatbed plotter. |
| | MBITS | MKBYTES | MICRO | This subroutine takes 8 word array, IWORD; assumes integers in IWORD are ASCII coded representations of '0', '1', or 'X'. Thus IWORD array represents binary coded microword (with X's standing for "Don't Care" bits. The subroutine translates X's into 0's, and returns in '12BYTES', the 16-bit value represented by the characters in the array IWORD. |
| | MINV -- Matrix Inversion | NONE | | The arguments are the dimensions (NDIM,NORDER) of the matrix, the matrix itself (it must all be in core), the determinant, and a work vector of dimensions (NDIM,2). Compile with the X switch for double precision. |
| | MHEX (NUMB, ISUM) | MKBYTES | | This program is changed from original version of MHEX in that it assumes four, rather than six hex digits, and assumes they are in the form 'XXXX'. This subroutine takes 3 ASCII coded words in array NUMB and returns in ISUM the decimal integer that the 3 words represent. It is assumed that the 6 characters in the 3 words represent four hexadecimal digits. IREGIS defines how those four digits are packed in the three words. There are two possibilities: "-X XX X-" and "-- XX XX". IREGIS=0 signals the first pattern, IREGIS=1 signals the second. Any character out of range is assumed to represent '0'. |
| | MHEX (NUMB,ISUM, IREGIS) | MKBYTES | MICRO | Subroutine that takes 3 ASCII coded words in array NUMB and returns in ISUM the decimal integer that the 3 words represent. It is assumed that the 6 characters in the 3 words represent four hexadecimal digits. IREGIS defines how those four digits are packed in the three words. There are two possibilities: "-X XX X-" and "-- XX XX". IREGIS=0 signals the first pattern, IREGIS=1 signals the second. Any character out of range is assumed to represent '0'. |
| | MKBYTES (M,IH, IL) | NONE | MHEX, MBITS | Program takes 16 bit word, 'M', assumes it is an ASCII representation of a pair of hexadecimal (or decimal) digits. It returns in integers 'IH' and 'IL' the numbers (0 to 15) that these digits represent. If any ASCII character is out of range, a 0 is returned for that character. A one is returned for 'X'. |
| | MMULT | NONE | ALG2.FR | All matrices must have integer elements. NOTE: Current version takes integer detector vector, multiplies it by integer matrix and stores in integer image array. Even though arithmetic is done floating point, there is a software check for any intermediate result overflowing double precision integer. |
| | MTP | NONE | | This is a transpose for large matrices stored on disk. It reads an input matrix from disk and writes the transpose to disk in channels designated by the calling sequence. The routine assumes that matrices are stored on disk with the first 2 integer words giving dimensions (rows first) and data following (row index varying fastest). Either read or double precision will work (X switch on compile for double) -- but not mixed files without dimensions at the beginning will also work as long as the calling routine positions channels at the start of the data area. |
| | NAMFIL | CFIL,DECOASC, ERROR | | Subroutine to create a disk file for data taking. The file name consists of 10 unique characters. The first letter is a 'T' for SOURCE=1 and a 'B' for SOURCE=2, an 'R' for SOURCE=3, an 'S' for SOURCE=4. The date follows (month and day). Next is a delimiting S. Finally the time (hour and minutes) is included. Hence: NXXXXSYYYY, where N is 'B', 'T', 'R' or 'S', XXXX is the date and YYYY is the time. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | NCON | NONE | | Subroutine to convert from DGDAC units to integer values. |
| | NUTLPS | NONE | Several programs incl. ILIPSS & CELL in SIMOSK, SMOSK | This subroutine generates the coordinates of the nutation pattern. X and Y coordinates ae generated parametrically as a function of theta as theta varies between zero and two-pi. |
| | OVLAP | JNPT | CELL in SIMOSK, SMOSK | This subroutine calculates the proper parameters so that CELL can calculate the amount of power in the image that falls on each quad cell. |
| | PGP | PGPDF, FIND | IMAGE, CIMAGE, FSIMAGE | Version of PGP with movie-making option, called by programs FSIMAGE, CIMAGE. This is a subroutine to display 64-pizel psuedo-color coded image on Genisco color system monitor. Includes color key and labeling. |
| | PGP | PGPDR, FIND | IMAGE, CIMAGE, FSIMAGE | Subroutine to display 64-pixel psuedo-color coded image on Genisco color system monitor. Includes color key and labeling. |
| | PGPIDM.SR | NONE | | This program is exactly the same as PGPDM, but with its name changed to PGPIDM. This was done to allow this program to be called in a program which calls another routine named PGPDR in a different overlay. |
| | PGPDR | NONE | | This version is for a non-memory-mapped environment. Use PGPDM in a memory-mapped environment. Subroutine PGPDR is an interface between a Nova RDOS program, written in FORTRAN, and the Genisco GCT3 operating system. It talks to the Genisco programmable graphics processor. It includes the following user callable routines:<br>DFPGP -- Defines the PGP to RDOS<br>RMPGP -- Removes the PGP from RDOS<br>WRPGP -- Starts a write of graphics file to PGP from Nova<br>WCPGP -- Continues write of graphics file to PGP from Nova<br>RDPGP -- Reads data from PGP into Nova memory. |
| | PGPDM.SR | NONE | | Does not work in non-memory-mapped environment. Must use PGPDR in non-memory-mapped environment. Listing of driver for GCT-3000 PGP. Subroutine PGPDR is an interface betwee a Nova RDOS program, written in FORTRAN, and the Genisco GCT3 operating system. It talks to the Genisco programmable graphics processor. It includes the following user callable routines:<br>DFPGP -- Defines the PGP to RDOS<br>RMPGP -- Removes the PGP from RDOS<br>WRPGP -- Starts a write of graphics file to PGP from Nova<br>WCPGP -- Continues write of graphics file to PGP from Nova<br>RDPGP -- Reads data from PGP into Nova memory. |
| | [illegible] | [illegible] version [illegible] | IMAGE | Subroutine to display 64-pixel psuedo-color coded image on Genisco color system monitor. This program does not include color key and labeling. This is because it is streamlined to run at maximum speed. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | ESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | PGPINIT | PGPOR,FIND | | Subroutine to initialize stat. of PGP to be ready to receive 7-raster plots of images |
| | PHEAD | NONE | HEADGN,SIMDSK, SMDSK,IMAGE, IMGSW | Subroutine to print header for disk file package used with AIT simulation study. Header array is passed in IHEAD and must be 80 words long. |
| | PLAYBK | PGPOR,FIND | IMAGE | Subroutine to display images on Genisco color system monitor rapidly in motion picture-like fashion. |
| | PLOT | NONE | | General purpose TTY plot subroutine NCURVE specifies the number of curves to plot (4=max). NPOINT points are plotted from the arrays 'X' and 'Y1,Y2,Y3,Y4'. NSCALE=0 specifies scaling done by this routine. If NSCALE=1, YMIN and YMAX from the calling routine will be used. A title ('TITLE') is printed of NTITLE characters. The plotting character (if 0) defaults to an asterisk (*). It may also be specified by the user. The plot may be done for an 80 or a 132 column. Device ('NCOL') errors are returned in IER. |
| | PLOTW | PLOT | IMAGE,IMGSW | Routine to plot the 4 time waveforms for AIT image [illegible]construction program. They are plotted on CRT or TTY. |
| | PMPIO | NONE | | This program was written in an attempt to solve a communications problem between the PMP and the Nova, having to do with 'CONTROL-S' and 'CONTROL-Q' (See program ASSDIAG.) This program sends the required one byte prompt to the PMP via a ULM line. This program then receives the two bytes from the PMP via the same ULM line. It assembles the bytes into a w[illegible] and passes the word back to the FORTRAN calling routine. The number of bytes receiv[illegible]nds on whether the PMP is signalling the detection of a memory error, or the end of a memory pass. In the former case, more information must be received from the PMP. This program obtains that information before control is returned to the calling routine. The ULM line channel is opened in the calling routine. This creates a problem, since the FORTRAN channel number does not correspond to the number used by the machine language as the channel number. This problem was solved by opening the ULM line in the calling routine in a certain order among all the channel opening instructions. |
| | PMPSHIP | CKSM | | This subroutine is called by most programs that communicate with the PMP. (Those written after this subroutine was written, anyway.) Passed the following four arguments: (1) a command code, (2) the number of words in the outgoing data packet, (3) the FORTRAN I/O channel on which outgoing communication to the PMP has been opened, and (4) an array containing the data packet to be shipped. This program will calculate the required checksum, and will s[illegible] the communication packet to the PMP using the present format desired by the microcode. This format is:<br>Argument 1: The command code (ICOM)<br>Argument 2: The 2's complement of the number of words in the data packet (-NWORDS)<br>Argument 3: The data packet (contents of IBUF)<br>Argument 4: A checksum, consisting of the two's complement of the lowest order 16 bits of the sum of all the previous information in the packet sent to the PMP. |
| | [illegible] | [illegible] | | [illegible] in program AITSES. It is called by [illegible]. It is passed a [illegible] and calculates the amount of power a unit point source would cause to be [illegible] the [illegible] spanned by the [illegible] (FWR[1-2]). It also determines [illegible] the crossing of the axes (MID1,MID2) |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | POWFMAT | NONE | | This program is a version of program POWTRK to be used with program FMATTRK. (It has its own output to file, 'PRINTER'.) For further documentation, see subroutine POWTRK and program FMATTRK. |
| | POWTRK FR | NONE | | This program determines the integrated power read from each detector during an arc of an AIT SCS tracker cycle. The input is from a bench derived or simulated detector vector. |
| | PRINTW | NONE | IMAGE, IMOSK, FSIMAGE | Routine to print time waveforms read from disk for AIT program. They may be printed on the CRT or the TTY printer. |
| | PTRK | NONE | | Special purpose version of subroutine POWTRK for use with subroutine TRK, whichis used in programs FSIMAGE, CIMAGE. This routine differs from POWTRK only in the fact that there is no common storage declared; all parameters are passed. This is for compatibility with image program structure. See POWTRK for further documentation. |
| | QCELL | CELL | QVECT in SIMDSK, SMDSK | This subroutine exists in order to "Call" CELL the correct number of times: once for each quad cell, and to provide a pointer to the place in the QC array for the current quad cell. |
| | QCGEN | INRI | ICUBE in SIMDSK, SMDSK | This subroutine creates the QC array which is a representation of the spatial [illegible]angement of the quad cells. Essentially, it does this by storing the minimum and maxim[illegible] [illegible] values for each Y coordinate in each quad cell, as well as storing the total maximum and minimum X and Y values for each quad cell. |
| | QVECT | QCELL & INRI | ICUBE in SIMDSK & SMDSK | This routine calls QCELL once for each time step in a nutation cycle and sets a pointer to the proper place in the output array for QCELL (actually, CELL) to put the calc[illegible] [illegible] waveform value. This routine also passes the nutation index, IPHI. |
| | FUNCTION RANF (SEED) | NONE | | Pseudo-random number generator linear congruential method is used with constants selected for 32 bit architecture. |
| | RDDATA | ERRCK | | Subroutine to read data from a file. Routine assumes 160 byte header preceding data. |
| | RDHEAD | ERRCK, PHEAD | | Subroutine to read the disk header. The header data is then printed. |
| | REPORT | WRBIN | | This subroutine was used in debugging communication problems between the Nova and the PMP that concerned the ULM-5 board. This program, given two 16-bit numbers, would print to the terminal (channel 10) in an easily to read form, a binary representation of the numbers. In this case, the 16-bit numbers represented status words from the Data General ULM driver, and easy to read means the fields of these 16-bit words were displayed in logical groupings. The subroutine WRBIN returns a 16 element array of ones and zeros, given a 16-bit number. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | | | | |
| | SCS.FR | POWER | | This program is called by AITSCS. Using the subroutine 'POWER', it determines the amount of power detected in each quadrant during a given tracker cycle, assuming the nutation of a point source. It also determines the point at which the source crosses the axes. The normalized X and Y ratios are then calculated and returned to the calling program. Code is provided for outputting to file 'PRINTER' various intermediate results. |
| | SCSFMAT | POWFMAT | | This program is a version of program SCSTRK to be used with program FMATTRK. (It has its own output to file 'PRINTER'.) For further documentation, see programs SCSTRK,FMATTRK. |
| | SCSGOF | POWER | | This program is a special purpose version of SCS.FR used by the program AITGOF. This program is not likely to be used again, but is being saved anyway. For further documentation, see AITGOF and SCS. |
| | SCSTRK | POWER (POWTRK) | | This subroutine determines the X and Y tracker ratios for a given tracker cycle (ITR), and optionally writes some intermediate values out to file 'PRINTER'. (This option is usually commented out.) The data file buffer (IBUF) is assumed to contain a detector vector, obtained from the bench or from simulation. |
| | SDMULT | NONE | | Subroutine SDMULT is matrix multiplication routine. It is a special version of MMULT to take single precision input and produce double precision output. |
| | SNDWD (IDATA) | TOUTA | | Uses serial I/O format to send 16-bit data and control words to the AIT nutation controller through a routine that addresses the Data General "DGDAC" Digital to Analogue Converter |
| | STRK | PTRK | | Special purpose version of subroutine SCSTRK for use with subroutine TRK, which is used in programs FSIMAGE, CIMAGE. This routine differs from SCSTRK only in the fact that there is no common, all parameters are passed. This is for compatibility with image program structure. See SCSTRK for further documentation. |
| | TINA | NONE | | Routine to handle TTL/5VLT input device on Data General Data Acquisition and Control subsystem without Sensory Access Manager software package. Routine is FORTRAN IV callable |
| | [illegible] | | | The variables in DATA common block below are provided by PLTWY. The meaning of other variables may be obtained from Sue Landon's memo of 11/14/79 or the terminal plot package |
| | [illegible] | NONE | | Routine to handle Data General Data Access and Control subsystem TTL output device without Sensory Access Manager software package. Routine is FORTRAN IV callable. |
| | [illegible] | [illegible] | | Special purpose version of program AITTRK used in program, CIMAGE, FSIMAGE. This routine differs from AITTRK only in the fact that there is no common, all parameters are passed. This [illegible] compatibility with IMAGE program structure. See AITTRK for further documentation. |

| FILE NAME | PROGRAM NAME | CALLS | CALLED BY | DESCRIPTION |
|---|---|---|---|---|
| SUBROUTINES | ULMRD.SR | NONE | | This program is used in communicating with PMP microcode. The protocol for the PMP is, each to be sent to the Nova is prompted by a one-byte communication from the Nova (value of the byte is not relevant). This program returns three integer parameters to the FORTRAN calling routine. The first parameter is the integer passed by the PMP. The next two are status words from the ULM-board, one for each byte. The ULM-5 documentation can be used in interpreting them as error situations arise. |
| | WAITGRAPH | WAITLOOP, PGPDR,FIND | | Version of GENGRAPH with a wait parameter which enables user to plot graphs in slow motion |
| | WAITLOOP | NONE | | This program introduces a wait of approximately one second for each increment of ten in the parameter 'INDEX'. |
| | WAITLOOP | NONE | | This program causes a delay each time it is called. The extent of the delay depends on the index parameter. Each increment of 10 in index introduces about [illegible] second of delay. |
| | WRBIN | NONE | | This program receives an integer in variable 'IMP'. It then puts into array IOUT ones and zeros corresponding to the binary representation of the integer IMP. |
| | WRDATA | ERRCK | | Subroutine to append data to a file with a header. NDATA is the number of words. IDATA is the address of the data. NAMDIR is the directory name. NAMFIL is the file name. IDEV is the device number. A '0' will select a 15 as the device number. IER is the error return. |
| | WRHEAD | ERRCK | | Subroutine to write header to disk file NAMFIL on directory NAMDIR. Device number IDEV is used |
| | WRHEX | NONE | | This program receives an integer in variable IMP. It translates this integer into ASCII hex representation, which it places in array IOUT. IOUT should be dimensioned to at least two. |

| addresses | number of copies |
|---|---|
| Capt. Patrick J. Martone<br>RADC/OCSE | 15 |
| RADC/TSLD<br>GRIFFISS AFB NY 13441 | 1 |
| RADC/DAP<br>GRIFFISS AFB NY 13441 | 2 |
| ADMINISTRATOR<br>DEF TECH INF CTR<br>ATTN: DTIC-DDA<br>CAMERON STA BG 5<br>ALEXANDRIA VA 22314 | 12 |
| Adaptive Optics Associates<br>Attn: Dr. L. E. Schmutz<br>2336 Massachusetts Avenue<br>Cambridge, Massachusetts 02140 | 5 |
| AFWL/ARAO<br>Attn: Dr. William Lowrey<br>Kirtland AFB, NM 87117 | 1 |
| AFWL/ARAA<br>Attn: Dr. J. Fender<br>Kirtland AFB, NM 87117 | 1 |

AFWL/ARAS 1
Attn: Lt Col Paul Bovenkirk
Kirtland AFB, NM 87117

The Aerospace Corp 1
Attn: Dr. E.W. Silvertooth
Bldg 110 MS 2339
PO Box 92957
Los Angeles, CA 90009

The Aerospace Corp 1
Attn: I. Taylor
Bldg 115 Room 13340
PO Box 92957
Los Angeles, CA 90009

Analytic Decisions Inc 1
Attn: Emmanuel Goldstein
1401 Wilson Blvd
Arlington, VA 22209

BDM Corp 1
Attn: William Gurley
1820 Randolph Rd
Albuquqrque, NM

BMD/ATC 1
Attn: W. Carmichael
PO Box 1500
Huntsville, AL 35807

Boeing Aerospace Co. 1
Attn: D. Allasina
PO Box 3999
Seattle, WA 98124

Charles Starke Draper Labs 1
Attn: Frank Scammel
555 Technology Sq.
Cambridge, MA 02139

Charles Starke Draper Labs 1
Attn: Dr. Keto Soosar
555 Technology Dr
MS 95
Cambridge, MA 02139

Corning Glass Works 1
Attn: E.T. Decker
Technical Products Division
Corning, NY 14830

DARPA/DEO 2
Attn: Col Ronald Prater
Arlington, VA 22209

Eastman Kodak 1
Attn: Robert Keim
Kodak Aparatus Division
121 Lincoln Ave.
Rochester, NY 14650

Eastman Kodak 1
Attn: Richard Price
KAD-Lincoln Park
901 Elmgrove Rd
Rochester, NY 14650

GRC 1
Attn: G. Gurski
7655 Old Springhouse RD
McLean, VA 22102

Hughes Aircraft 1
Attn: Martin Flannery
MS E/125 Bldg 6
Centenela & Teal Sts
Culver City, CA 90230

Hughes Aircraft 1
Attn: Dr Fred McClung
MS6-E 125
Centenela & Teal Sts
Culver City, CA 90230

Itek Corp 1
Attn: Roland Plante
Optical Systems Division
10 Maguire Rd.
Lexington, MA 02173

Lockheed Palo Alto Research Lab 1
Attn: Richard Feaster
0/52-03, B201
3251 Hanover St.
Palo Alto, CA 94304

Lockheed Space and Missile Co. 1
Attn: Dennis Aspinwall
Dept 5203 Bldg. 201
3251 Hanover St.
Palo Alto, CA 94304

Lockheed Space and Missile Co. 1
Attn: Dick Wallner
Dept 5203 Bldg 201
3251 Hanover St.
Palo Alto, CA 94304

Martin Marietta Aerospace
Attn: C.W. Spieth
Denver Division
PO Box 179
Denver, CO 80201

Denver Division 1
PO Box 179
Denver, CO80201

MIT/Lincoln Laboratory 1
Attn: Alex Parker
PO Box 73
Lexington, MA 02173

[illegible] Corp 1
Attn: Mr. Kenneth Robinson
71 Blake St.
Needham, MA 02192

| | |
|---|---|
| NASA Ames<br>Attn: James Murphy<br>MS 244-7<br>Moffett Field, CA 94035 | 1 |
| NASA Marshall Space Flight Center<br>Attn: Charles O. Jones<br>Mail Code EC32<br>Huntsville, AL 35812 | 1 |
| Naval Sea Systems Command<br>Attn: Dr. Sadeg Sianatgar<br>PMS-405<br>NC 1 Room 11N08<br>Washington, DC 20742 | 1 |
| Naval Weapons Center<br>Attn: Dr. H. Bennett<br>Code 0018<br>China Lake , CA 93555 | 1 |
| Perkin Elmer<br>Attn: Dr. David Dean<br>MS 241<br>Main Ave<br>Norwalk, CT 06856 | 1 |
| Perkin Elmer<br>Attn: Henry Dieselmen<br>100 Wooster Heights Rd.<br>Danbury, CT 06810 | 1 |
| Rockwell International<br>Attn: R. Brandenie<br>Rocketdyne Division<br>6633 Canoga Ave<br>Canoga Park, CA 91304 | 1 |
| Rockwell International<br>Attn: J. Murphy<br>Space Division<br>12214 Lakewood Blvd<br>Downey, CA 90241 | 1 |

Rockwell International 1
Attn: R. Greenberg
Space Division
12214 Lakewood Blvd
Downey, CA 90241

Riverside Research Institute 1
Attn: Dr. Robert Kappesser
1701 N Fort Myer Dr.
Suite 711
Arlington, VA 22209

SD/YNS 1
Attn: Col H.A. Shelton
PO Box 92960
Worldway Postal Center
Los Angeles, CA 90009

United Technologies Research Center 1
Attn: Dr. J. Pearson
Optics & Applied Technology Lab
PO Box 2691
West Palm Beach, FL 33402

University of Arizona 1
Attn: Prof Robert Shannon
&Charles Peyton
Administration Bldg
Tucson, AZ 85721

W.J. Schafer Assoc. Inc. 1
Attn: Edward Borsare
10 Lakeside Office Park
Wakefield, MA 01880